Django a Python web framework

Sommaire

- Présentation de Django
- Avantages
- Inconvénients
- Exemple d’utilisation
- Sujet de TP

I/ Présentation

- Python : langage interprété permettant de développer en moins de ligne qu'un autre langage

- Django : framework open source et gratuit écrit en python

Philosophie de Django : la simplicité

Histoire de Django

- Créé en 2003
- Journal local : World Online
- Objectif : réduire considérablement le temps de développement
- Publié en 2005
- Rencontre un grand succès

II/ Avantages

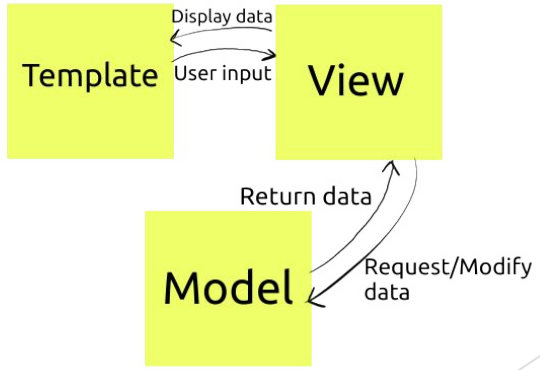
Pourquoi utiliser Django ?

- Simplicité d'apprentissage
- Maintenance
- Rapidité de développement
- Sécurité
- Qualité des applications
- Gestion de gros flux de données

Facilité d'apprentissage et maintenance

- Utilisation de Python
- Langage facile à lire
- Utilisation de gabarits

- Prise en charge du MVT
- Séparation des couches



Rapidité de développement

Utilisation d'un ORM

```python
from django.db import models

class Produit(models.Model):
    nomProduit = models.CharField(primary_key=True, max_length=50)

    def __str__(self):
        return self.nomProduit
```

1 classe = 1 table = 1 model

1 propriété = 1 colonne

ORM (Object-Relational Mapping)

Migration vers n'importe quel base (Postgres, SQlite, etc)

Méthodes prédéfinies (SELECT, DELETE, etc)

Rapidité de développement

Utilisation d'un ORM, jointure

```python
from django.contrib.auth.models import User
from django.db import models

class Produit(models.Model):
    nomProduit = models.CharField(max_length=50)
    user = utilisateur = models.ForeignKey(User, on_delete=models.CASCADE)

    def __str__(self):
        return self.nomProduit
```

Si pas de clé primaire django crée automatique un id qui sera la clé primaire

Rapidité de développement

Migration BDD

```python
from django.db import models

class Produit(models.Model):
    nomProduit = models.CharField(primary_key=True, max_length=50)

    def __str__(self):
        return self.nomProduit
```



```sql
create table produit(
    nomProduit varchar(50) not null primary key
);
```

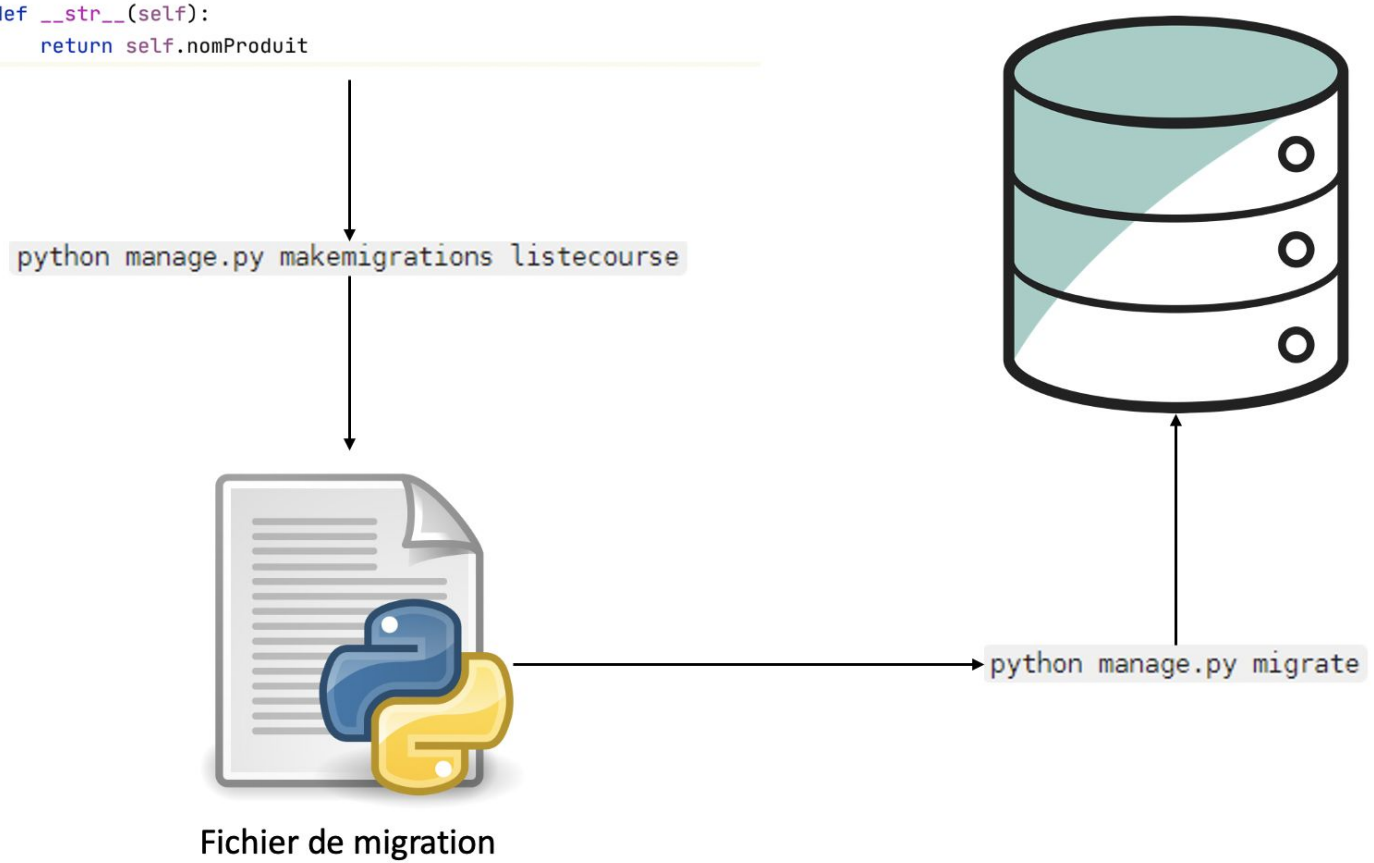
Rapidité de développement

Migration BDD

```python
from django.db import models

class Produit(models.Model):
    nomProduit = models.CharField(primary_key=True, max_length=50)

    def __str__(self):
        return self.nomProduit
```



Rapidité de développement

Conception formulaire

```python
from django.http import HttpResponseRedirect
from django.shortcuts import render
from django.urls import reverse
from django import forms


def index(request):
    if request.method == 'POST':
        form = LoginForm(request.POST)
        if form.is_valid():
            username = request.POST['username']
            password = request.POST['password']
            # Ajouter les actions à faire ici
            return HttpResponseRedirect(reverse('accueil'))
    else:
        form = LoginForm()

    return render(request, 'listecourse/login.html', {'form': form})


class LoginForm(forms.Form):
    username = forms.CharField(label='Nom d\'utilisateur', max_length=100)
    password = forms.CharField(widget=forms.PasswordInput(), label='Mot de passe')
```

Rapidité de développement

Conception formulaire

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Connexion</title>
</head>
<body>
    <form action="" method="POST">
        {% csrf_token %}
        {{form.as_ul}}
        <input type="submit" value="Connexion"/>
    </form>
</body>
</html>
```

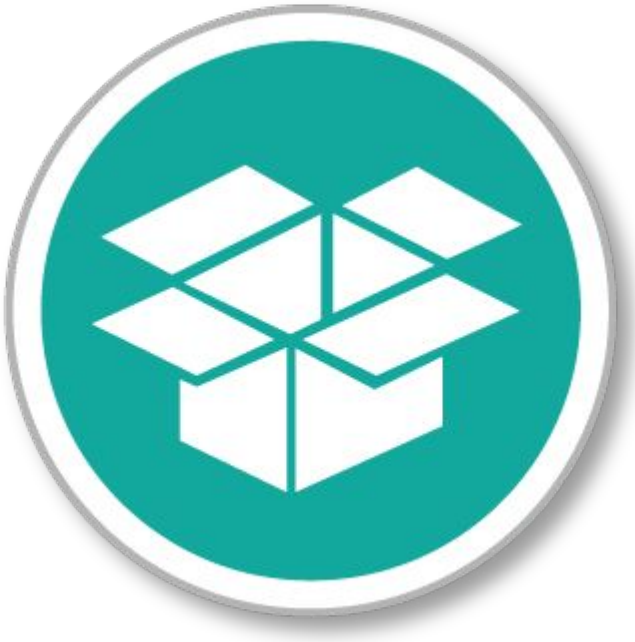
Rapidité de développement

Conception formulaire

```
<form action="" method="POST">
  <li>
    <label for="id_username">Nom d&#39;utilisateur :</label>
    <input type="text" name="username" maxlength="100" required id="id_username">
  </li>
  <li>
    <label for="id_password">Mot de passe :</label>
    <input type="password" name="password" required id="id_password">
  </li>
  <input type="submit" value="Connexion"/>
</form>
```

Rapidité de développement

Packages fournis et partageables



Rapidité de développement

Panneau d'administration (destiné aux gestionnaires)

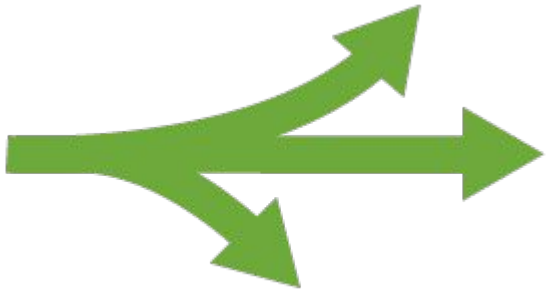
Pour créer un utilisateur admin :

```
python manage.py createsuperuser
```

Sécurité

- Cross-Site Request Forgery
- Cross Site Scripting
- Injections SQL
- ClickJacking

Flux de données



- Performant dans la gestion des flux importants

- Gestion flux RSS
- Facilité d'implémentation

Flux de données

Téléversement de fichier

< 2.5 Mo

- Mise en mémoire
- Simple lecture et écriture

- Ecriture dans un fichier temporaire
- Ecriture progressive du fichier

> 2.5 Mo

Exemple téléversement fichier

forms.py

```python
from django import forms

class UploadFileForm(forms.Form):
    title = forms.CharField(max_length=50)
    file = forms.FileField()
```

Définition du formulaire avec une classe

Champs FileField

Si aucune requête , récupération du formulaire pour la vue

Si le formulaire est envoyé :

- vérification de la validité
- gestion de l'upload

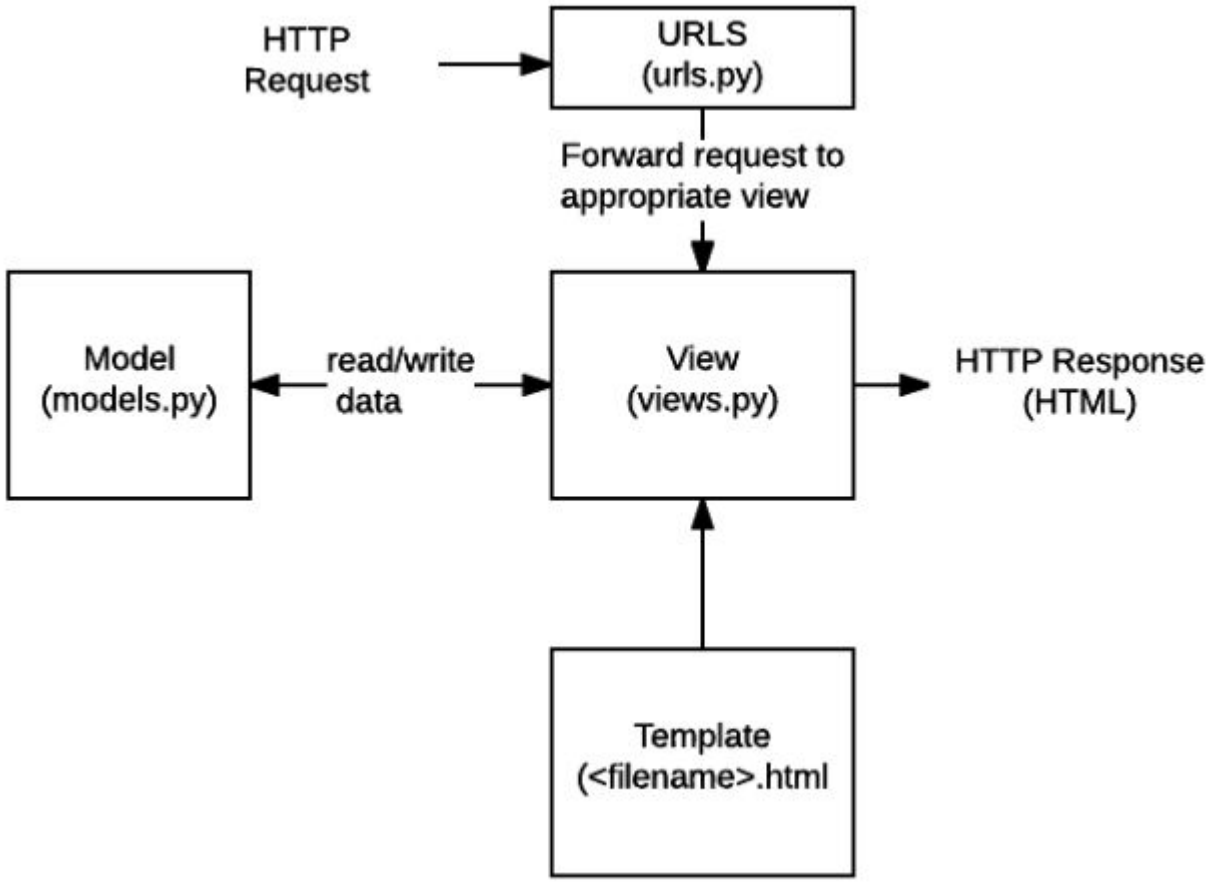
views.py

```python
from django.http import HttpResponseRedirect
from django.shortcuts import render
from .forms import UploadFileForm

# Imaginary function to handle an uploaded file.
from somewhere import handle_uploaded_file

def upload_file(request):
    if request.method == 'POST':
        form = UploadFileForm(request.POST, request.FILES)
        if form.is_valid():
            handle_uploaded_file(request.FILES['file'])
            return HttpResponseRedirect('/success/url/')
    else:
        form = UploadFileForm()
    return render(request, 'upload.html', {'form': form})
```

Model View Template



III/ Inconvénients

Inconvénients

- Pas fait pour les petits projets
- Manque de conventions
- Configuration complexe
- Ne gère pas les requêtes simultanées
- Prise en main compliquée

IV/ Comparaison avec Tornado

Django VS Tornado

- Meilleure documentation
- Panneau d’administration
- Django possède un ORM
- Puissance de flux
- Beaucoup de packages existants et grande communauté

IV/ Exemples d’utilisation

Applications

Vive les mariés !

V/ Complément technique

Les URLs

```
urlpatterns = [
    path('', accueil.index, name='accueil'),
    path('accounts/login/', login.index, name='login'),
    path('logout', logout.index, name='logout'),
    path('registration', registration.index, name='registration'),
    path('addProduct', addProduct.index, name='addProduct'),
    path('addProductInList', addProductInList.index, name='addProductInList'),
    path('deleteProductList/<str:nom_produit>', accueil.deleteProduit, name='deleteProductList'),
    path('moreQuantite/<str:nom_produit>', accueil.moreQuantite, name='moreQuantite'),
    path('lessQuantite/<str:nom_produit>', accueil.lessQuantite, name='lessQuantite'),
]
```

Documentation URL

Authentification d'un USER

```
from django.contrib.auth import authenticate, login
username = request.POST['username']
password = request.POST['password']
user = authenticate(request, username=username, password=password)
if user is not None:
    login(request, user)
```

Documentation utilisateur

Déconnexion d'un USER

```
from django.contrib.auth import logout
def index(request) :
    logout(request)
```

Création d'un USER

```
from django.contrib.auth.models import User
from django.contrib.auth import authenticate,
user = User.objects.create_user(username, email, password)
user.save()
```

Méthode sécurisée

```python
from django.contrib.auth.decorators import login_required
from django.contrib.auth.models import User
from django.shortcuts import render
@login_required
def index(request):
    context = {'liste': User.objects.all()}
    return render(request, 'gestion/accueil.html', context)
```

Si l'utilisateur n'est pas connecté l'annotation renvoie à l'URL : `accounts/login/`

Interroger la base de données

Récupérer toutes les lignes d'une table :

```
from django.contrib.auth.models import User
 user = User.objects.all()
```

Ajouter un filtre à la requête: (peut retourner plusieurs lignes)

```
from django.contrib.auth.models import User
 userList = User.objects.filter(email="toto@gmail.com")
```

Pour récupérer un unique objet dans la base de données : (Si plusieurs ou zéro lignes, déclenchement d'une exception) :

```
from django.contrib.auth.models import User
user = User.objects.get(username="toto")
```

Pour supprimer un objet en base de données :

```
from django.contrib.auth.models import User
 user = User.objects.get(username="toto")
 user.delete()
```

Documentation requête

Utilisation du MVT

utilisateurs.py

```
from django.contrib.auth.models import User
from django.shortcuts import render

def index(request):
    userlist = User.objects.all()
    context = {"userlist" : userlist}
    return render(request, 'user.html', context)
```

Utilisation du MVT

templates/user.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Utilisateurs</title>
</head>
<body>
{% for user in userlist %}
    <p>{{ user.username }}</p>
{% endfor %}

</body>
</html>
```

Utilisation du MVT

urls.py

```
from django.urls import path

from .views import utilisateurs

urlpatterns = [
    path('utilisateurs',utilisateurs.index ,name='utilisateur')
```

Utilisation du MVT

admin

VI/ Sujet de TP

Conception d'une application

- Réalisation d'une application de gestion d'une liste de courses

https://gitlab-etu.fil.univ-lille1.fr/couture/ari-django