Web avancé et Micro-services

Remaci Zeyneb Yassmina

Python

- Python est un langage de programmation puissant et facile à apprendre.
- Il possède des structures de données avancées.
- Il permet une programmation orientée objet de façon simple et efficace.
- Sa syntaxe claire et son typage dynamique le rendent agréable à utiliser.

Framework web

•Un Framework signifie littéralement « cadre de travail ».

•C’est un ensemble structuré de composants logiciels, outils et conventions.

•Il fournit une base pour créer des applications logicielles.

•Il simplifie le développement en proposant des solutions pré-conçues aux problèmes courants.

Micro-framework web

- Un micro-framework est un ensemble minimal de modules pour développer des applications web rapidement.
- Il fournit les fonctionnalités courantes nécessaires au développement.
- Lors de la création d'une application web, il permet de gérer :
  - les requêtes HTTP,
  - le serveur web,
  - l'affichage de pages web dynamiques,
  - les cookies, etc.
- L'objectif est de simplifier et accélérer le développement tout en restant léger et flexible.

Flask

•Développé par Armin Ronacher en 2010, pour offrir un cadre minimal et léger.

•Citation de l'auteur : *« Le but de Flask est de donner une bonne base de travail pour toute application. Le reste ne dépend que de vous ou d'extensions. »*

•Micro-framework de python: offre les fonctionnalités de base nécessaires au développement d'une application web.

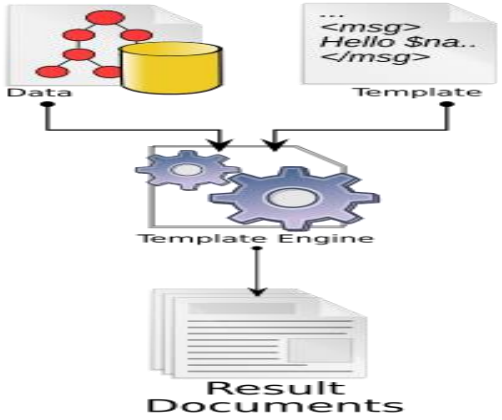
•Il permet de concevoir des applications web professionnelles de façon simple et rapide.

•Utilisé par LinkedIn en interne et pour l'API de Pinterest, ainsi que par de nombreux sites à fort trafic.

•Popularité : puissant et léger, permet de créer une application web minimale en seulement 7 lignes.

Moteur de modèles (Template engine)

- Un moteur de template (également appelé moteur de modèles ou parseur de modèles) est un logiciel conçu pour combiner des modèles avec un modèle de données afin de produire des documents finaux·



Moteur de templates Jinja2

•Jinja2 est un moteur de templates pour Python.

•Il sert à générer du texte ou des fichiers dynamiques à partir de modèles.

•Il est utilisé pour créer pages web, fichiers de configuration, documentation ou rapports.

Installation Flask

1. **Installer VS code**
2. **Vérifier l'installation de Python: python3 –version / py –v**
3. python3 -m venv venv
4. .\flaskapp\Scripts\Activate.ps1
5. pip install Flask
6. python -m flask –version
7. Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser

1er pas

```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
@app.route("/about")
def about():
    return "<i>Bonjour</i>"
```

Lancer l’application

- $env:FLASK_APP = "myapp.py"
- flask run

2 eme methode:

- py myapp.py

```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
@app.route("/about")
def about():
    return "<i>Bonjour</i>"


if __name__ == "__main__":
    app.run(debug=True)
```

```python
from flask import Flask,  render_template, url_for
```

```python
@app.route("/about")
def about():
    return render_template("test.html")
```

```
events = [
    {
        'title': 'Conférence sur l'IA et la santé',
        'category': 'Intelligence Artificielle',
        'organizer': 'Omar',
        'thumbnail': 'ai_health.jpg'
    },
    {
        'title': 'Atelier développement web',
        'category': 'Développement Web',
        'organizer': 'Remaci',
        'thumbnail': 'web_workshop.jpg'
    },
    {
        'title': 'Séminaire sur la cybersécurité',
        'category': 'Sécurité Informatique'
```

```python
@app.route("/about")
def about():
    return render_template("test.html",events=events)
```

```
<html lang="en">
<head>
</head>
<body>
    {% for event in events %}
     <div class="card-body text-center">
                                <h4 class="card-title mb-4">
                                    {{event.title}}
                                </h4>
                                <p class="card-text">{{event.category}}</p>
                            </div>
      {%endfor%}
</body>
</html>
```

```python
@app.route("/about")
def about():
    return render_template("test.html",events=events,title="TP1")
```

```
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   {% if title %}
   <title>{{title}}</title>
   {% else %}
   <title>Mon TP</title>
   {%endif%}

</head>
```

```
<html lang="en">
<head>
    <title>{{title}}</title>
     {% else %}
    <title>Mon TP</title>
    {%endif%}

</head>
<body>

     {% block content %}

       {%endblock%}
</body>
</html>
```

```
{%extends "test2.html"%}

{%block content%}
    {% for event in events %}
     <div class="card-body text-center">
                                <h4 class="card-title mb-4">
                                    {{event.title}}
                                </h4>
                                <p class="card-text">{{event.category}}</p>
                            </div>
      {%endfor%}

      {%endblock content%}
```

url_for

```
{%extends "test2.html"%}

{%block content%}
    {% for event in events %}
     <div class="card-body text-center">
        <img src="{{url_for('static', filename=event.icon)}}" alt="" >
                                    <h4 class="card-title mb-4">
                                        {{event.title}}
                                    </h4>
                                    <p class="card-text">{{event.category}}</p>
                                </div>
      {%endfor%}

     {%endblock content%}
```

url_for

- <link rel="stylesheet" type="text/css" href="{{url_for('static',filename='main.css')}}"/>

```
class RegistrationForm(FlaskForm):
    fname = StringField('First name',
                        validators=[DataRequired(), Length(min=2,max=30)])
    lname = StringField('Last name',
                        validators=[DataRequired(), Length(min=2,max=30)])
    username = StringField('user name', validators=[DataRequired(), Length
    email = StringField('email',
                        validators=[DataRequired(), Email()])
    pwd = PasswordField('Password',
                        validators=[DataRequired(),
                                    Regexp( "^(?=.*[a-z])(?=.*[A-Z])(?=.*
    confirm = PasswordField('PConfirm pwd', validators=[DataRequired(),Equ
    submit = SubmitField("Sign up")
```

```
from flask import app
from flask_wtf import FlaskForm
from wtforms import BooleanField, StringField, PasswordField,
SubmitField
from wtforms.validators import DataRequired, Length, Email,
Regexp , EqualTo
```

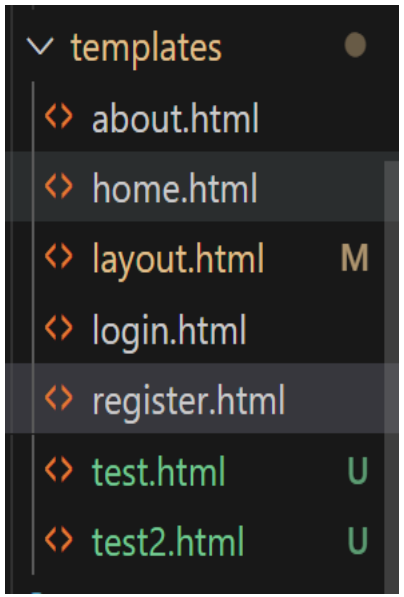
```
{%extends "layout.html"%}

{%block content%}
<div class="container mt-auto align-items-center">
    <form action="" method="POST">
        {{form.hidden_tag()}}
        <fieldset class="form-group">
            <legend class="border-bottom mb-4">Create an Account</legend>
            <div class="form-group">
                {{form.fname.label(class='form-control-label')}}
                {{form.fname(class='form-control form-control-lg')}}
            </div>
            <div class="form-group">
                {{form.lname.label(class='form-control-label')}}
                {{form.lname(class='form-control form-control-lg')}}
            </div>
```

```
    <div class="border-top pt-3">
        <small class="text-muted">
            Already have an account ?
            <a href="{{url_for('login')}}"  class="ml">Sign in</a>
        </small>
    </div>


</div>

{%endblock content%}
```



```python
from form import RegistrationForm , LoginForm
```

```python
@app.route("/register",methods=["GET","POST"])
def register():
    form = RegistrationForm()

    return render_template('register.html',title='Register',form = form)
```

TP1

Application de gestion de restaurants / commandes

1. Page profil.
2. Page Plats et catégories.
3. Page panier.
4. Page commande .

Application de suivi de santé / bien-être

- Page des médecins.
- page des patients.
- page des ordonnances.
- Page médicament.

Plateforme de blog collaboratif

- Page Article.
- Page commentaire.
- Page catégorie.
- Page évènement.

Plateforme de gestion de bibliothèque

- Page catégorie des livres.
- Page livre.
- Page emprunt.
- Page emprunteurs.

GIT

- **Système de contrôle de version distribué (DVC) :**
- Chaque développeur dispose d'une copie locale complète du dépôt, ce qui permet de travailler hors ligne.

GITLab

- GitLab est une plateforme DevOps et DevSecOps qui permet de gérer l'ensemble du cycle de vie du développement logiciel, de la conception du code à son déploiement en production. Il offre un espace collaboratif pour l'hébergement et le contrôle de versions de projets Git, intégrant des outils pour l'intégration et la livraison continues (CI/CD), la gestion des tâches, la sécurité, et le déploiement, le tout sur une plateforme unique.

GITLab

- Installer Git.
- git config --global user.name « Remaci »
- git config --global user.email « zeynebyasmina.remaci@univ-tlemcen.dz«
- cd /chemin/vers/ton/projet
- git init
- git add .
- git commit -m "Premier commit«

GITLab

- Va sur gitlab
- Cliquez sur le bouton new projet
- Cliquez sur le bouton create blank project
- Copie l’URL du dépôt
- Lier ton dépôt local au dépôt GitLab : git remote add origin https://gitlab.com/username/mon-projet.git
- Pousser votre projet sur GitLab:
  - git branch -M main
  - git push -u origin main

- git checkout -b ma-branche
- git push -u origin ma-branche