

FLASK



Objective

À la fin de ce cours, vous saurez

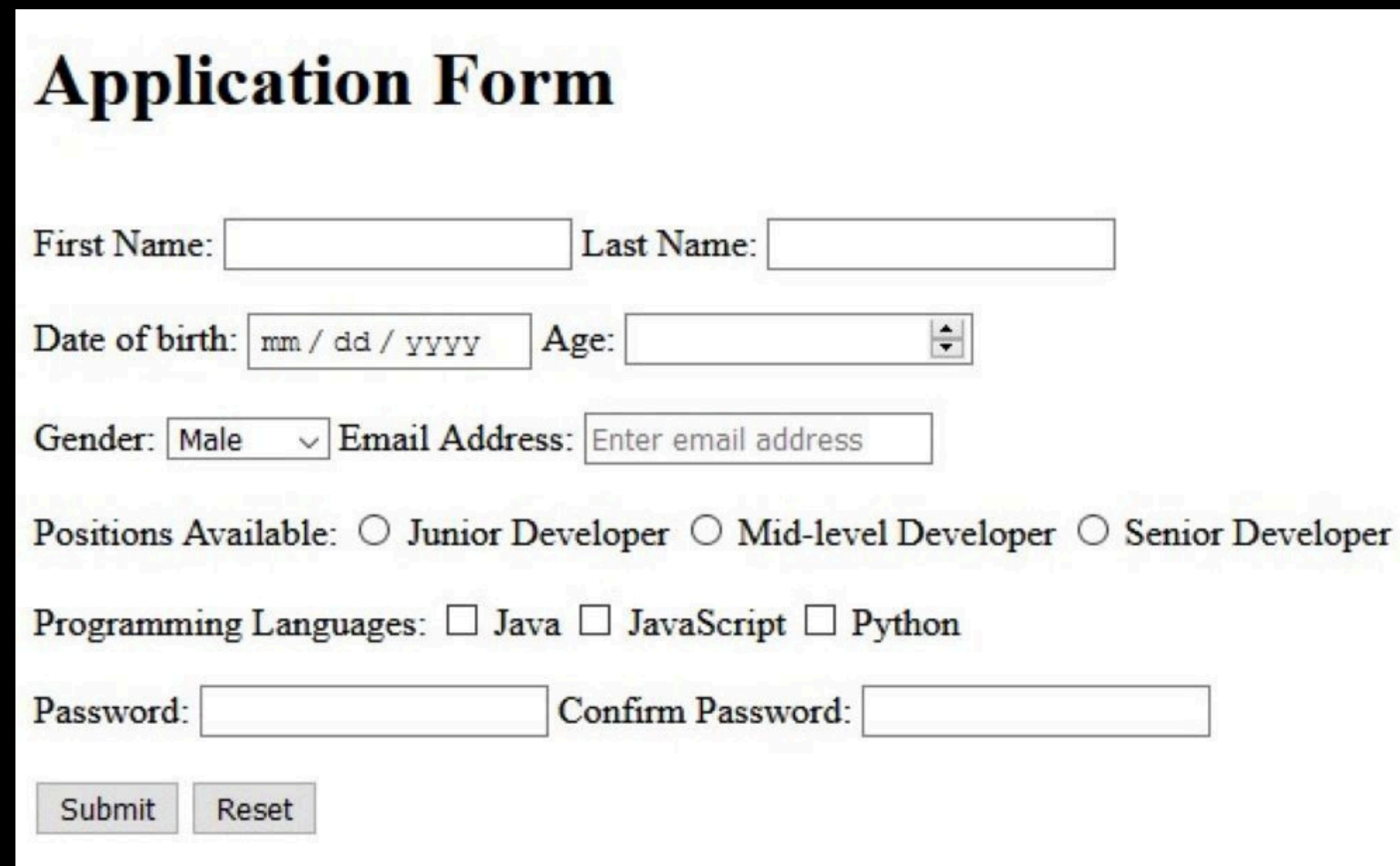
- Comment gérer les données de formulaire dans Flask en utilisant la requête (request)
- Faire la distinction entre les méthodes POST et GET
- Utiliser les sessions et les Cookies pour stocker les données utilisateur
- Interagir avec une base de données SQLite.

En général, nous allons donc nous concentrer aujourd'hui sur le fait de considérer tout ce qui se trouve après le nom de domaine comme un chemin, en général. Et un synonyme de cela, dans le contexte du Web, serait également un itinéraire. Ainsi, un itinéraire se résume à un certain nombre de lettres, peut-être quelques barres obliques, peut-être une extension de fichier qui fait référence à une partie de votre application.

Request.Form

Qu'est-ce que le concept de formulaires dans le développement web ?

Dans le développement web, un formulaire est un moyen pour les utilisateurs d'interagir avec une application web en fournissant des données de saisie. Les formulaires sont utilisés pour collecter les données de l'utilisateur, telles que du texte, des nombres, des dates et des fichiers, et les envoyer au serveur pour traitement.



Application Form

First Name: Last Name:

Date of birth: Age:

Gender: Email Address:

Positions Available: ☐ Junior Developer ☐ Mid-level Developer ☐ Senior Developer

Programming Languages: ☐ Java ☐ JavaScript ☐ Python

Password: Confirm Password:

Les formulaires peuvent être utilisés à diverses fins, notamment :

- User registration and login
- Contact forms
- Surveys and feedback forms
- Payment processing
- File uploads

Comment Request.Form semble-t-il ?

Request.form est un objet similaire à un dictionnaire dans Flask qui contient les données du formulaire envoyées dans la requête HTTP. C'est un moyen d'accéder aux données que l'utilisateur a saisies dans un formulaire et soumises au serveur.

```
> ImmutableMultiDict([('username', 'john_doe'), ('email', 'john@example.com')])
```

POST and GET Methods

Quels sont les méthodes HTTP ?

HTTP(Protocole de transfert hypertexte) est un protocole utilisé pour transférer des données sur Internet. Les méthodes HTTP, également appelées méthodes de requête.

1. **GET**: Récupérer une ressource
2. **POST**: Créer une nouvelle ressource
3. **PUT**: Mettre à jour une ressource existante
4. **DELETE**: Supprimer une ressource

En résumé,

GET : est utilisé pour récupérer des données.

POST : est utilisé pour créer ou mettre à jour des données.

- La méthode POST est généralement utilisée pour envoyer des données au serveur.

Sessions

imagine que tu te connectes à un site de shopping en utilisant ton nom d'utilisateur et ton mot de passe. Une fois connecté, le site crée une session pour toi.

Sessions : "Je garde tes informations importantes avec moi (serveur) pendant que tu es sur mon site, mais dès que tu pars, je les efface."

Le concept de session

- Stocker des données utilisateur de manière temporaire pendant une session de navigation.
- Une session est une période de temps pendant laquelle un utilisateur interagit avec un site web ou une application.

1. Authentifier les utilisateurs

2. Suivi des activités

3. personnaliser l'expérience
utilisateur

- Banque en ligne

- Réseaux sociaux

- E-commerce

Cookies

Imagine que tu vas sur un site de shopping en ligne, et que ce site te demande de choisir ta langue préférée (français ou anglais). Une fois que tu choisis, le site enregistre cette préférence dans un cookie sur ton ordinateur.

Cookies : "Je te laisse un petit post-it sur ton bureau (ordinateur) pour me souvenir de ce que tu aimes."

Le concept de Cookies

- Mécanisme de stockage côté client qui permet aux serveurs web de conserver certaines informations sur l'utilisateur et de les récupérer lors des requêtes suivantes.
- **Request** pour lire les cookies et l'objet **response** pour définir des cookies.

1. langue préférée

2. Theme préférée (sombre ou clair)

3. personnaliser l'expérience
utilisateur

4. Authentifier les utilisateurs

- Site web en général

SQLite3

une base de données légère, intégrée et facile à utiliser, qui stocke les données dans un simple fichier. Contrairement à d'autres systèmes de gestion de bases de données comme MySQL ou PostgreSQL, SQLite n'a pas besoin d'un serveur dédié pour fonctionner. Tout se passe dans un fichier sur ton disque dur.

```
# Connexion à une base de données (si le fichier n'existe pas, il sera créé)  
conn = sqlite3.connect('contacts.db') cursor = conn.cursor()
```

```
conn.commit() # Sauvegarde les changements
```

```
conn.close() # Ferme la connexion
```