Praktikum - Erstellung einer einfachen Blog-Plattform mit Flask und SQLite

Dein Name

July 31, 2023

Schritt 1: Projektvorbereitung

Wir beginnen damit, ein neues Projekt zu erstellen. Hierzu legen wir einen Ordner für das Projekt an und benennen ihn beispielsweise "blog". In diesem Ordner erstellen wir eine Datei mit dem Namen **app.py**.

Schritt 2: Grundlegende Flask-Anwendung

- In der Datei **app.py** importieren wir die notwendigen Module: **Flask** aus Flask und **render_template** aus Flask.
- Dann initialisieren wir die Flask-Anwendung mit **app = Flask(__name__)**.
- Wir definieren die Wurzelroute ("/") mit einer einfachen Funktion **index()**, die "Willkommen auf meiner Blog-Plattform!" zurückgibt.
- Wir starten die Anwendung, indem wir **app.run(debug=True)** verwenden.

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return 'Willkommen auf meiner Blog-Plattform!'

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

Schritt 3: Datenbankmodell definieren

- Als nächstes werden wir eine Datenbank erstellen, um unsere Blog-Beiträge zu speichern. Wir nutzen dazu die Erweiterung SQLAlchemy.
- Importiere **SQLAlchemy** aus **flask_sqlalchemy**.
- Wir definieren unser Datenbankmodell **BlogPost**. Dieses Modell wird später eine Tabelle in der Datenbank darstellen. Es hat eine **id**, einen title und einen **content** für jeden Blog-Beitrag.

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///blog.db'
db = SQLAlchemy(app)

class BlogPost(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100), nullable=False)
    content = db.Column(db.Text, nullable=False)

    def __repr__(self):
        return f"<BlogPost␣{self.id}:␣{self.title}>"
```

Schritt 4: Datenbank erstellen und Beiträge anzeigen

- Wir erstellen die Datenbank mit **db.create_all()**. Dabei wird die Tabelle für die Blog-Beiträge in der Datenbank erstellt.
- In der Funktion **index()** rufen wir alle Blog-Beiträge aus der Datenbank ab und geben sie auf der Startseite aus. Dazu nutzen wir die Funktion **render_template()**, um eine HTML-Vorlage mit den Beiträgen anzuzeigen.

```
from flask import Flask, render_template
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///blog.db'
db = SQLAlchemy(app)

class BlogPost(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100), nullable=False)
    content = db.Column(db.Text, nullable=False)

    def __repr__(self):
        return f"<BlogPost␣{self.id}:␣{self.title}>"

@app.route('/')
def index():
    posts = BlogPost.query.all()
    return render_template('index.html', posts=posts)
```

Schritt 5: HTML-Vorlage für die Startseite

- Erstelle im gleichen Ordner wie **app.py** einen Unterordner namens "templates".
- In diesem Unterordner erstellst du eine Datei namens **index.html**.
- Die HTML-Vorlage wird die Blog-Beiträge auf der Startseite anzeigen. Wir nutzen dazu eine Schleife, um durch die Beiträge zu iterieren und sie anzuzeigen.

```
<!DOCTYPE html>
<html>

<head>
    <title>Mein Blog</title>
</head>

<body>
    <h1>Mein Blog</h1>
    <ul>
        {% for post in posts %}
        <li>
            <h2>{{ post.title }}</h2>
            <p>{{ post.content }}</p>
        </li>
        {% endfor %}
    </ul>
    <a href="{{ url_for('create') }}">Neuen Beitrag erstellen</a>
</body>

</html>
```

Schritt 6: Einen neuen Blog-Beitrag erstellen

- Wir wollen dem Benutzer die Möglichkeit geben, einen neuen Blog-Beitrag zu erstellen. Hierfür fügen wir eine neue Route "/create" hinzu.
- Wenn der Benutzer ein Formular absendet, speichern wir den neuen Beitrag in der Datenbank und leiten den Benutzer zurück zur Startseite.

```
from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///blog.db'
db = SQLAlchemy(app)

class BlogPost(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100), nullable=False)
    content = db.Column(db.Text, nullable=False)

    def __repr__(self):
        return f"<BlogPost␣{self.id}:␣{self.title}>"

@app.route('/')
def index():
    posts = BlogPost.query.all()
    return render_template('index.html', posts=posts)

@app.route('/create', methods=['GET', 'POST'])
def create():
    if request.method == 'POST':
        title = request.form['title']
        content = request.form['content']
        new_post = BlogPost(title=title, content=content)
        db.session.add(new_post)
        db.session.commit()
        return redirect(url_for('index'))
    return render_template('create.html')

if __name__ == '__main__':
    with app.app_context():
        db.create_all()
    app.run(debug=True)
```

Schritt 7: HTML-Vorlage für das Erstellen eines Beitrags

- Erstelle im "templates" Unterordner eine Datei namens **create.html**.
- Diese Vorlage wird ein Formular anzeigen, in das der Benutzer den Titel und Inhalt des neuen Blog-Beitrags eingeben kann.

```
<!DOCTYPE html>
<html>

<head>
    <title>Neuen Beitrag erstellen</title>
</head>

<body>
    <h1>Neuen Beitrag erstellen</h1>
    <form method="POST" action="{{ url_for('create') }}">
        <label for="title">Titel:</label>
        <input type="text" id="title" name="title" required><br>
        <label for="content">Inhalt:</label>
        <textarea id="content" name="content" rows="4"
            required></textarea><br>
        <button type="submit">Speichern</button>
    </form>
    <a href="{{ url_for('index') }}">Zur Startseite</a>
</body>

</html>
```

Herzlichen Glückwunsch! Du hast jetzt eine einfache Blog-Plattform mit Flask und einer SQLite-Datenbank erstellt. Die Startseite zeigt alle vorhandenen Blog-Beiträge an und über die "/create"-Route kannst du neue Beiträge erstellen und in der Datenbank speichern.