Come gestire l'invio di un form in POST con Flask in Python

Flask è un microframework leggero e flessibile per lo sviluppo web in Python. Una delle funzionalità fondamentali offerte da Flask è la gestione dei moduli (forms) inviati tramite il metodo HTTP POST. In questo articolo, vedremo come creare, inviare e gestire un form utilizzando Flask.

Prima di iniziare, assicurati di avere Flask installato nel tuo ambiente Python. Puoi installarlo utilizzando pip:

```
pip install Flask
```

Inizia creando una semplice applicazione Flask. Crea una cartella per il progetto e all'interno un file chiamato `app.py`:

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

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/submit', methods=['POST'])
def submit():
    if request.method == 'POST':
```

```
        name = request.form['name']
        email = request.form['email']
        # Aggiungi ulteriori elaborazioni o
salvataggio dei dati nel database
        return f"Received: Name - {name}, Email -
{email}"

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

Nella stessa directory, crea una cartella chiamata `templates` e all'interno un file `index.html`:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form Submission</title>
</head>
<body>
    <form action="{{ url_for('submit') }}"
method="post">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name"><br>
<br>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email">
<br><br>
        <input type="submit" value="Submit">
    </form>
```

```
</body>
</html>
```

Ossia:

- **Creazione dell'Applicazione Flask**: L'applicazione Flask viene creata e configurata nel file `app.py`.
- **Definizione delle Rotte**: La funzione `index()` gestisce la rotta di base (`/`) e renderizza il template HTML contenente il form. La funzione `submit()` gestisce la rotta `/submit` e viene chiamata quando il form viene inviato.
- **Gestione della Richiesta POST**: La funzione `submit()` verifica se il metodo della richiesta è POST, estrae i dati inviati dal form utilizzando `request.form`, e li elabora (ad esempio, stampandoli o salvandoli in un database).

Per avviare l'applicazione, esegui il file `app.py`:

```
python app.py
```

Accedi al tuo browser e vai all'indirizzo `http://127.0.0.1:5000/` per visualizzare il form. Compila i campi e invia il form per vedere i dati elaborati dal server.

Possiamo a questo punto aggiungere le seguenti feature:

- **Validazione dei Dati**: È importante validare i dati ricevuti per garantire che siano corretti e sicuri. Flask-WTF è una libreria che può aiutare con la validazione dei moduli.
- **Salvataggio nel Database**: Puoi estendere l'applicazione per salvare i dati in un database utilizzando SQLAlchemy o altre librerie di gestione del database.

- **Gestione degli Errori**: Implementa una gestione degli errori appropriata per gestire eventuali problemi durante l'elaborazione del form.

Ecco un esempio di come integrare Flask-WTF per la validazione dei moduli:

```
from flask import Flask, render_template, request,
redirect, url_for, flash
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField
from wtforms.validators import DataRequired, Email

app = Flask(__name__)
app.secret_key = 'supersecretkey'

class MyForm(FlaskForm):
    name = StringField('Name', validators=
[DataRequired()])
    email = StringField('Email', validators=
[DataRequired(), Email()])
    submit = SubmitField('Submit')

@app.route('/', methods=['GET', 'POST'])
def index():
    form = MyForm()
    if form.validate_on_submit():
        name = form.name.data
        email = form.email.data
        flash(f'Received: Name - {name}, Email -
{email}')
        return redirect(url_for('index'))
    return render_template('index.html', form=form)
```

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

Nel template `index.html`, utilizza Flask-WTF per generare il form:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form Submission</title>
</head>
<body>
    <form method="post" action="">
        {{ form.hidden_tag() }}
        <p>
            {{ form.name.label }}<br>
            {{ form.name(size=32) }}<br>
            {% for error in form.name.errors %}
                <span style="color: red;">[{{ error
}}]</span><br>
            {% endfor %}
        </p>
        <p>
            {{ form.email.label }}<br>
            {{ form.email(size=32) }}<br>
            {% for error in form.email.errors %}
                <span style="color: red;">[{{ error
}}]</span><br>
            {% endfor %}
        </p>
        <p>{{ form.submit() }}</p>
```

```
    </form>
    {% with messages = get_flashed_messages() %}
        {% if messages %}
            <ul>
                {% for message in messages %}
                    <li>{{ message }}</li>
                {% endfor %}
            </ul>
        {% endif %}
    {% endwith %}
</body>
</html>
```

Conclusione

Gestire l'invio di un form tramite POST in Flask è relativamente semplice e può essere arricchito con validazione e gestione degli errori utilizzando librerie come Flask-WTF. Con queste conoscenze, puoi sviluppare applicazioni web più interattive e sicure.