LU06d - Routing und Variable Rules in Flask

In Flask ist das Routing ein zentrales Konzept, das bestimmt, wie URLs an bestimmte Funktionen gebunden werden. Jede Funktion ist mit einer oder mehreren URLs verknüpft, sodass, wenn eine dieser URLs angefordert wird, die zugehörige Funktion aufgerufen wird. Mit variablen Regeln kann Flask dynamische URLs verarbeiten, die Teile enthalten, die sich ändern.

Grundlegendes Routing

In Flask wird das Routing mit dem `@app.route()`-Dekorator durchgeführt. Zum Beispiel:

```
@app.route('/')
def home():
    return 'Homepage'
```

In diesem Beispiel wird die Funktion `home()` aufgerufen, wenn die Haupt-URL (`/`) der Anwendung angefordert wird.

Variable Rules

Mit Flask können Sie spezifische Abschnitte einer URL als Variablen definieren. Diese Variablen werden dann an die zugehörige Funktion als Argumente übergeben.

```
@app.route('/user/<username>')
def show_user_profile(username):
    return f'User {username}'
```

In diesem Beispiel wird die `show_user_profile()`-Funktion aufgerufen, wenn eine URL wie `/user/john` angefordert wird. Der Wert `john` wird dann an die Funktion als `username`-Argument übergeben.

Es ist auch möglich, den Datentyp der Variablen zu definieren:

```
@app.route('/post/<int:post_id>')
def show_post(post_id):
    return f'Post {post_id}'
```

Hier wird erwartet, dass `post_id` ein Integer ist. Wenn die URL `/post/123` angefordert wird, wird die `show_post()`-Funktion mit `post_id` als 123 aufgerufen.

Unique URLs / Redirection Behavior

Flask hat eine interessante Eigenschaft, wenn es um die Endung von URLs geht. Es unterscheidet

zwischen URLs mit und ohne abschließendem Schrägstrich.

```
@app.route('/about')
def about():
    return 'About Page'
```

Wenn du `/about/` (mit einem abschließenden Schrägstrich) besuchst, wird Flask automatisch auf `/about` (ohne Schrägstrich) umleiten. Dies stellt sicher, dass URLs eindeutig sind, was hilft, doppelte Inhalte zu vermeiden und die Suchmaschinenoptimierung (SEO) zu verbessern.

Umgekehrt, wenn du eine Route mit einem abschließenden Schrägstrich definierst, wird Flask den Benutzer zu dieser URL mit Schrägstrich umleiten, wenn sie ohne besucht wird.

```
@app.route('/projects/')
def projects():
    return 'Projects Page'
```

Ein Besuch von `/projects` wird automatisch zu `/projects/` umgeleitet.

HTTP Methods

Standardmäßig reagieren die in Flask definierten Routen nur auf GET-Anfragen. Wenn du jedoch andere HTTP-Methoden wie POST, PUT oder DELETE verarbeiten möchtest, musst du dies explizit angeben.

```
from flask import request

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        return 'Login via POST'
    else:
        return 'Login via GET'
```

In diesem Beispiel reagiert die `login()`-Funktion sowohl auf GET- als auch auf POST-Anfragen. Mit `request.method` kannst du überprüfen, welche HTTP-Methode verwendet wurde, und entsprechend reagieren.

M323-LU06