Flask Documentation
by
Pinkesh Mahawar

February 16, 2024

1 Installation

- Flask supports the latest version of Python above 3.8.
- Automatically installed dependencies:
  1. Werkzeug: implements WSGI, interface between applications and servers.
  2. Jinja: template language that renders the pages your application serves.
  3. Click: framework for writing command line applications.
- Create virtual environment:
  - In Windows:

```
mkdir myproject
cd myproject
py -3 -m venv .venv
```

  - In Linux:

```
mkdir myproject
cd myproject
python3 -m venv .venv
```

- Activate the environment:
  - In Windows:

```
.venv\Scripts\activate
```

- In Linux:

```
. .venv/bin/activate
```

- Install Flask (can install it globally):

```
pip install Flask
```

2 Minimal Application

- Format looks like this (`hello.py` with `.py` extension):

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
```

- **Foundation/Initialization:** Python imports Flask as a class and creates an instance of this class using `__name__`. This is needed to look for resources such as templates and static files. The `route()` decorator tells Flask to call the function on that particular URL. With this function, you can return a message to display in the user's browser. It may be content in HTML or a simple message.

- **Run Application:**

```
flask --app hello run

    When you go to the URL \url{http
       ↪ ://127.0.0.1:5000} or \url{http
       ↪ ://127.0.0.1:5000/}, you can see \texttt
       ↪ {Hello, World!} on the screen in HTML
       ↪ format within a \texttt{<p>} tag.
    \item Example of more routes:
    \begin{lstlisting}
@app.route('/welcome')
def hello():
    return 'Welcome back!'

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

This is one way to pass variables.

- Example with request handling:

```
from flask import request

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

`GET` and `POST` requests are used to access data. `GET` requests are not secure for sending data because the data can be seen in the URL like `http://example.com/path/to/resource?param1=value1param2=value2param3=value3`. Therefore, it's prioritized to use `POST` requests to send data.

- Example with template rendering:

```
from flask import render_template

@app.route('/hello/<name>')
def hello(name=None):
    return render_template('hello.html', name=name
        ↪ )
```

`hello.html` template:

```
<!doctype html>
<title>Hello from Flask</title>
{% if name %}
    <h1>Hello {{ name }}!</h1>
{% else %}
    <h1>Hello, World!</h1>
{% endif %}
```

We can use Jinja2 to access variables from the server to show in an HTML page. It behaves like Python in logic, where logic is worked within {% `logic` %} and variables are used as {{`var`}} which are returned by the server via `render_template`.

- Example of redirecting users:

```
from flask import redirect, url_for

@app.route('/homepage')
def index():
    if user is not login:
        return redirect(url_for('login'))
    else:
        return render_template('hompage.html')
```

We can redirect a user to another URL if required.