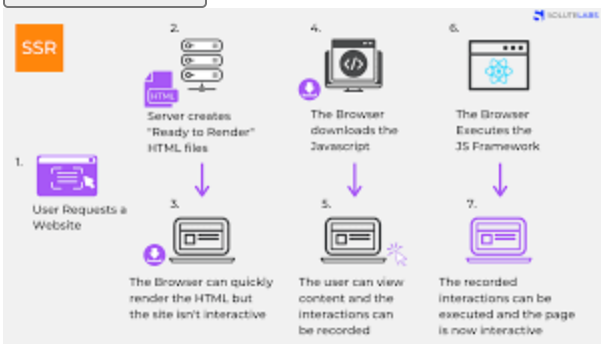
Generate PDF



How to Create a Simple Web Server with Flask

Introduction

Flask is a lightweight web framework for Python that allows you to build web applications quickly and efficiently. In this tutorial, you'll learn how to set up a simple web server using Flask and create a basic web application.

Prerequisites

To follow this tutorial, you will need:

- Basic knowledge of Python programming.
- Python installed on your computer (version 3.6 or later).
- A code editor (e.g., Visual Studio Code, PyCharm, or any text editor).

Step 1: Install Flask

1. Open your terminal or command prompt.
2. Install Flask using pip by running the following command:

```
bash

Copy code
pip install Flask
```

1. This will download and install the Flask framework and its dependencies.

Step 2: Create Your Project Folder

1. Create a new folder for your project, for example, `flask_app`.
2. Inside this folder, create a new Python file named `app.py`.

Step 3: Write the Flask Code

Open `app.py` in your text editor and add the following code:

```
python

Copy code
from flask import Flask, render_template

app = Flask(__name__)
```

```
@app.route('/')
def home():
    return "<h1>Welcome to My Flask Web Server!</h1><p>This is a simple web application.</p>"

@app.route('/about')
def about():
    return "<h1>About</h1><p>This web server was created using Flask.</p>"

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

Explanation of the Flask Code:

- Import the Flask class and the `render_template` function.
- Create an instance of the Flask class. This instance will be our WSGI application.
- Define routes using the `@app.route()` decorator. The `home()` function returns a welcome message, and the `about()` function returns information about the server.
- Finally, check if the script is being run directly and call `app.run()` to start the web server in debug mode, which is useful for development.

Step 4: Run Your Flask Application

1. In the terminal, navigate to your project folder:

```
bash

Copy code
cd path/to/flask_app
```

1. Run your Flask application with the following command:

```
bash

Copy code
python app.py
```

1. You should see output indicating that the server is running, typically at `http://127.0.0.1:5000/`.

Step 5: View Your Web Application

1. Open your web browser.
2. Go to `http://127.0.0.1:5000/` to see the welcome message.
3. Navigate to `http://127.0.0.1:5000/about` to see the about page.

Step 6: Customize Your Web Application

Feel free to modify the content in the `home()` and `about()` functions to customize your web application. You can also explore adding HTML templates and static files (like CSS and JavaScript) for more complex web applications.

Step 7: Adding Templates (Optional)

To use HTML templates, follow these steps:

1. Create a new folder named `templates` inside your project folder.
2. Inside the `templates` folder, create a new file named `home.html` and add the following code:

```
html
```

```
Copy code
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home</title>
</head>
<body>
    <h1>Welcome to My Flask Web Server!</h1>
    <p>This is a simple web application using templates.</p>
    <a href="/about">About</a>
</body>
</html>
```

1. Update the `home()` function in `app.py` to render this template:

```
python

Copy code
@app.route('/')
def home():
    return render_template('home.html')
```

1. Save the changes and refresh your browser at `http://127.0.0.1:5000/` to see the new home page.

Conclusion

Congratulations! You have successfully created a simple web server using Flask. You can now expand this project by adding more routes, templates, and features like forms, user authentication, and database integration. Flask provides the flexibility and power to build robust web applications, so continue exploring and experimenting!