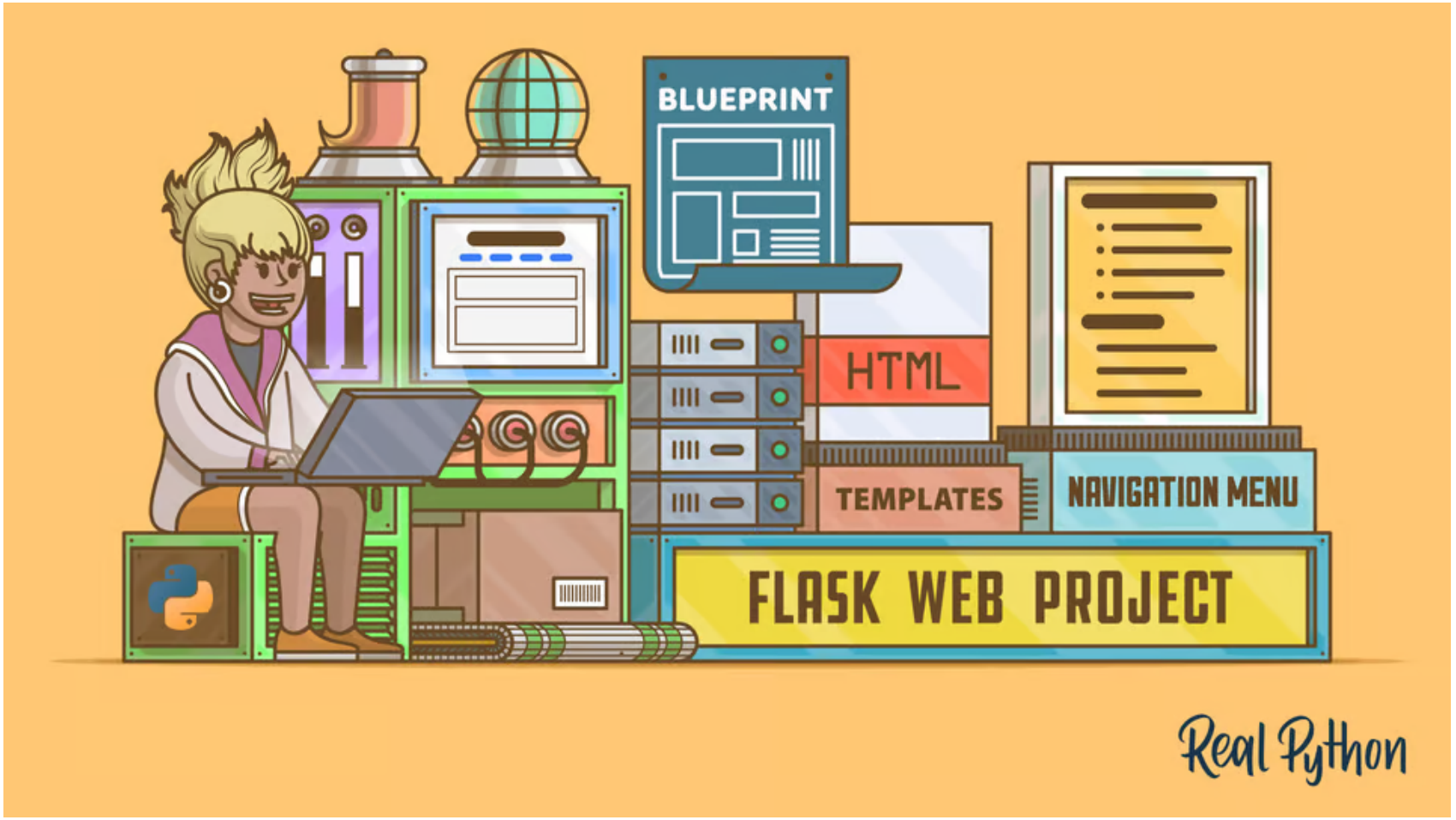
Build a Scalable Flask Web Project From Scratch



Course Overview

- Project Overview
- Create Your First Flask Project
- Restructure Your Project for Scalability
- Leverage Blueprints and Templates
- Handle the Navigation
- Enhance the UI
- Conclusion

Project Overview

1. **What you'll build**
2. What you'll learn

Project Overview

1. What you'll build
2. **What you'll learn**

What You'll Learn

- How to build a Flask project from scratch

What You'll Learn

- How to build a Flask project from scratch
- Structure your Flask project for scalability

What You'll Learn

- How to build a Flask project from scratch
- Structure your Flask project for scalability
- Understand the flow of requests and responses

What You'll Learn

- How to build a Flask project from scratch
- Structure your Flask project for scalability
- Understand the flow of requests and responses
- Manage the application using packages

What You'll Learn

- How to build a Flask project from scratch
- Structure your Flask project for scalability
- Understand the flow of requests and responses
- Manage the application using packages
- Improve the user experience and UI

Create Your First Flask Project

1. **Prerequisites**
2. Set up the environment for your Flask project
3. Create the Flask project
4. Run the Flask project

Prerequisites

- Python 3.9 or greater
- An IDE or Code Editor (VS Code / PyCharm)

Create Your First Flask Project

1. Prerequisites
2. **Set up the environment for your Flask project**
3. Create the Flask project
4. Run the Flask project

Set Up the Environment for Your Flask Project

- Why use virtual environments?
- Create and activate your virtual environment
- Install the required dependencies in your virtual environment

Create Your First Flask Project

1. Prerequisites
2. Set up the environment for your Flask project
3. **Create the Flask project**
4. Run the Flask project

Create the Flask Project

- Import the Flask library:

```
from flask import Flask
```

- Create an instance of the `Flask` class:

```
app = Flask(__name__)
```

- Define a route for your homepage:

```
@app.route("/")
def index():
    return "Hello, World!"
```

Create Your First Flask Project

1. Prerequisites
2. Set up the environment for your Flask project
3. Create the Flask project
4. **Run the Flask project**

Restructure Your Project for Scalability

- Create a package folder named `board/`
- Move `app.py` into `board/`
- Rename `app.py` to `__init__.py`
- Edit `__init__.py` to define the Application Factory

Leverage Blueprints and Templates

1. **Utilize blueprints**
2. Create the templates

Utilize Blueprints

- Why use blueprints?
- Create blueprints
- Register blueprints with your Flask project

Leverage Blueprints and Templates

1. Utilize blueprints
2. **Create templates and add them to a blueprint**
3. Render the templates

Create Templates and Add Them to a Blueprint

- Jinja templates
- Build the base template
- Add the child templates

Leverage Blueprints and Templates

1. Utilize blueprints
2. Create templates and add them to a blueprint
3. **Render the templates**

Render the Templates

- Import the `render_template` function
- Return the template

Handle the Navigation

1. **Create the navigation template**
2. Embed the navigation template in the base template

Create the Navigation Template

- Why create a separate navigation template?
- Create a separate template for the navigation bar
- Add links to other pages

Handle the Navigation

1. Create the navigation template
2. **Embed the navigation template in the base template**

Embed the Navigation Template in the Base Template

- Use the `{% include %}` tag in the base template

Enhance the UI

1. **Structure your project for static files**
2. Style your project using CSS

Structure Your Project for Static Files

- Create a `static/` folder at the root level of your project
- Use the `url_for()` function to generate URLs for static files

Enhance the UI

1. Structure your project for static files
2. **Style your project using CSS**

Conclusion

1. **What you learned**
2. Continue learning
3. Additional resources

What You Learned

- Fundamentals of Flask web development
- Best practices for structuring a scalable Flask project
- Using blueprints and templates for modularity
- Handling the navigation and the user interactions
- Enhancing the user interface with CSS

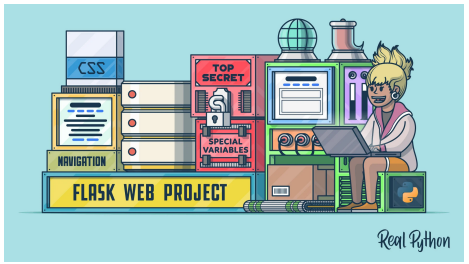
Conclusion

1. What you learned
2. **Continue learning**
3. Additional resources

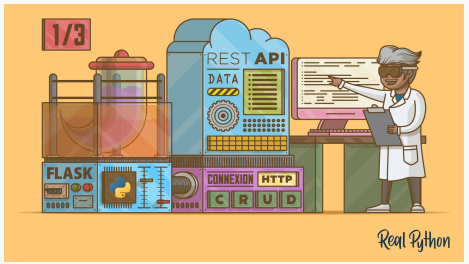
Continue Learning

- Database integration to store data
- User authentication and authorization
- API development with Flask-RESTful
- Deployment on a web server

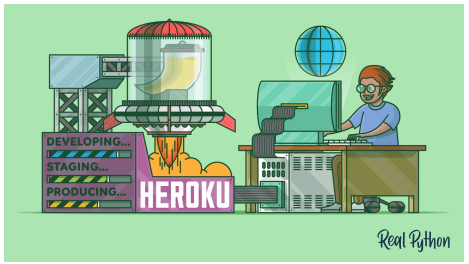
Additional Resources



Enhance Your Flask Web Project With a Database

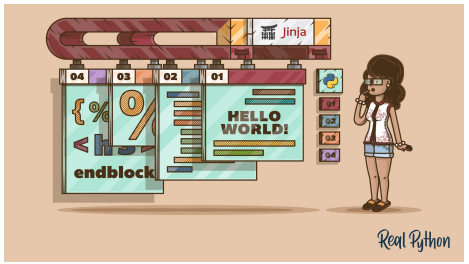


Python REST APIs with Flask, and SQLAlchemy

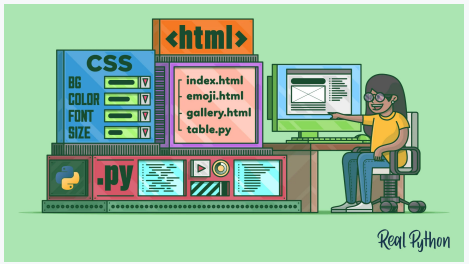


Deploying a Python Flask Application on Heroku

Additional Resources



Jinja Templating



HTML and CSS for Python Developers