Austin WEBpython Presentation – Visible Flask – No Surprises

This is not a tutorial. I list multiple tutorials with different styles of development, different package support, and different scope in the Resources section below.

My experience with the tutorials was that I did not get any real understanding of “how” Flask worked – or even what it was worth to me – during the course of following the tutorial. Also, I had trouble evaluating the approaches of the different tutorials to determine which would work best for me. No complaints. The authors were doing their best to disclose their experience and guide me to a point where I could exploit Flask effectively. But, because of my own learning patterns, I was frequently frustrated – and frequently lost – as I worked my way through the examples.

Most problems stem from my ignorance of the details Flask smooths over for us. I did not understand enough about the HTTP transaction protocols and pitfalls. I did not understand the history of Apache’s mod\_python or WSGI. I still have not mastered a full, secure, multi-threaded web application with an Nginx webserver and Postgres database backend. Maybe I’ll get there by the end of this series.

Definition: EXPERT – someone who has done it more times than you have. Usually that means once.

I have done all of this stuff at least once. Maybe I have done some of them more times than you have.

If you don’t “get” what I just showed you, ask me to do it AGAIN – or ASK ME TO DO IT A DIFFERENT WAY. (Limit: two, maybe three, ways.)

BACKTRACKING is OK. Ask me to go back a few steps to a previous example.

Maybe not “Best Practices.” But “Good” or at least “Workable” practices + some deeper understanding that will help guide you to “Better Practices.”

Yak Shaving: You want to do A. But in order to do A, you have to learn B. But you cannot learn B without understanding C. And you cannot do C until you have installed D. And D depends on E. And before you can get E, you have to know … Yak Shaving.

The exercise we are about to undertake is not Yak Shaving, though it will sometimes have that feel. Most of what we will see and do can be presented in a tutorial without any real explanation – monkey see, monkey do style.

DEMO: PyCharm – Single file Flask Hello World. (Discussion)

DEMO: PyCharm – Application package Hello World. (Discussion)

DISCUSSION: Virtual Environments

DEMO: Command Line:

Show python and python3 system installations.

Create python3 virtual environment.

Fail to run demos with virtual environment.

Pip install Flask in virtual environment.

Check out dir(app)

Check out dir(app\_context)

Resources:

Grinberg – [Mega-Tutorial](https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world)

Grinberg – [Flask Web Development ...](https://www.amazon.com/Flask-Web-Development-Developing-Applications/dp/1449372627/ref=sr_1_1?ie=UTF8&qid=1488052543&sr=8-1&keywords=miguel+grinberg)

[Creating a Web App From Scratch Using Python Flask and MySQL](https://code.tutsplus.com/tutorials/creating-a-web-app-from-scratch-using-python-flask-and-mysql--cms-22972)

.

.

.

Show simple example application.

Grinberg’s Hello World.

Some other Hello World.

Flask directory layout 🡪 project organization

Smoothing out the Request/Response shuttle.

Flask package is open source: <https://github.com/pallets/flask>

Website: <http://flask.pocoo.org>

What confused me:

* HTTP 🡪 Request/Response
* Route decorators on “view” functions
* Forms

Make a virtual environment; run the app; throw it away; remake; run the app

Rework the app into common, simple directory structure; run the app;

Fail with debug on; fail with debug off

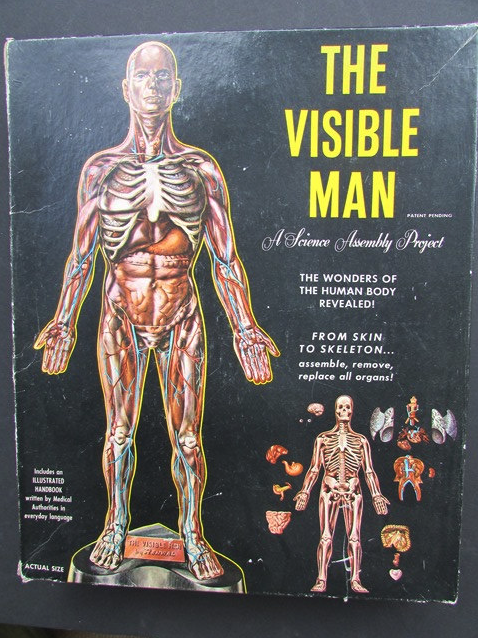
Pass information in URL; pass information in form; pass information in cookie

* **Part I. Introduction to Flask**
  + 1. Installation
    - Using Virtual Environments
    - Installing Python Packages with pip
  + 2. Basic Application Structure
    - Initialization
    - Routes and View Functions
    - Server Startup
    - A Complete Application
    - Flask Extensions
    - The Request-Response Cycle
  + 3. Templates
    - The Jinja2 Template Engine
    - Twitter Bootstrap Integration with Flask-Bootstrap
    - Custom Error Pages
    - Links
    - Static Files
    - Localization of Dates and Times with Flask-Moment
  + 4. Web Forms
    - Cross-Site Request Forgery (CSRF) Protection
    - Form Classes
    - HTML Rendering of Forms
    - Form Handling in View Functions
    - Redirects and User Sessions
    - Message Flashing
  + 5. Databases
    - SQL Databases
    - NoSQL Databases
    - SQL or NoSQL?
    - Python Database Frameworks
    - Database Management with Flask-SQLAlchemy
    - Database Use in View Functions
    - Integration with the Python Shell
    - Database Migrations with Flask-Migrate
  + 6. Emails
    - Email Support with Flask-Mail
  + 7. Large Application Structure
    - Project Structure
    - Configuration Options
    - Application Package
    - Launch Script
    - Unit Tests
    - Requirements File
    - Database Setup
* **Part II. Example: A Social Blogging Application**
  + 8. User Authentication
    - Authentication Extensions for Flask
    - Password Security
    - Creating an Authentication Blueprint
    - User Authentication with Flask-Login
    - New User Registration
    - Account Confirmation
    - Account Management
  + 9. User Roles
    - Database Representation of Roles
    - Role Assignment
    - Role Verification
  + 10. User Profiles
    - Profile Information
    - User Profile Page
    - Profile Editor
    - User Avatars
  + 11. Blog Posts
    - Blog Post Submission and Display
    - Blog Posts in Profile Pages
    - Paginating Long Lists of Blog Posts
    - Rich Text Posts with Markdown and Flask-PageDown
    - Permanent Links to Blog Posts
    - Blog Post Editor
  + 12. Followers
    - Database Relationships Revisited
    - Followers in the Profile Page
    - Query Followed Posts using a Database Join
    - Show Followed Posts in the Home Page
  + 13. User Comments
    - Database Representation of Comments
    - Comment Submission and Display
    - Comment Moderation
  + 14. Application Programming Interfaces (APIs)
    - Introduction to REST
    - RESTful Web Services with Flask
* **Part III. The Last Mile**
  + 15. Testing
    - Obtaining Code Coverage Reports
    - The Flask Test Client
    - End-to-End Testing with Selenium
    - Is It Worth It?
  + 16. Performance
    - Logging Slow Database Performance
    - Source Code Profiling
  + 17. Deployment
    - Deployment Workflow
    - Logging of Errors In Production
    - Cloud Deployment
    - The Heroku Platform
    - Traditional Hosting
  + 18. Additional Resources
    - Using an Integrated Development Environment
    - Finding Flask Extensions

Getting Involved with Flask

= = = = = = = = = = = = = = = = = = = = = =

Not a Tutorial – The Visible Man



Examine the guts of Flask – from skin to skeleton.

Includes the subsystems or companion packages on which it relies.

Includes the development environment – directory structure, IDE, git, etc.

Look at internals: applications, contexts, …

Examine assumptions and options: directory names, file names, config values …

Add-on capabilities:

Database

Login

…

= = = = = = = = = = = = = = = = = = = = = =

Werkzeug

In the Box

* HTTP header parsing and dumping
* Easy to use request and response objects
* Interactive JavaScript based in-browser debugger
* 100% WSGI 1.0 compatible
* Supports Python 2.6, 2.7 and 3.3.
* Unicode support
* Basic session and signed cookie support
* URI and IRI utilities with Unicode awareness
* Built-in library of fixes for buggy WSGI servers and browsers

Integrated routing system for matching URLs to endpoints & vice versa

WSGI – Some pronounce as “whiskey” – hence Bottle, Flask

WSGI is Python standard for communication btw web server and local Python process

Werkzeug + Flask Application + itsdangerous == sanitizes HTTP communications

Werkzeug + Flask Application + itsdangerous == abstracts HTTP communications

= = = = = = = = = = = = = = = = = = = = = =

Jinja2

UNICODE

XSS

…

= = = = = = = = = = = = = = = = = = = = = =

itsdangerous

Communicating with untrusted source.

= = = = = = = = = = = = = = = = = = = = = =

Flask object

Dir()

Dir() with keyword groupings

= = = = = = = = = = = = = = = = = = = = = =

Application object

Dir()

= = = = = = = = = = = = = = = = = = = = = =

Routing

@appname.route(…)

must be imported to build routing table before appname.run() [[???]]

RESTful with parameters

= = = = = = = = = = = = = = = = = = = = = =

HTTP communications

Protocol

Request object

Response object

= = = = = = = = = = = = = = = = = = = = = =

Request object

Cookie

Form

URL

Get, Post, etc.

= = = = = = = = = = = = = = = = = = = = = =

= = = = = = = = = = = = = = = = = = = = = =

Response object

Essentially a string

= = = = = = = = = = = = = = = = = = = = = =

= = = = = = = = = = = = = = = = = = = = = =