Building a Web Application in Python Using Flask

Katherine Fogarty, Brandyn Price, and Ben Wynn

February 4th, 2023

Agenda

- About Peraton
- Flask Introductory Overview
- Flask Deep Dive
  - REST APIs
  - Request Object
  - Error Handling
  - Templates
  - Databases
- Demo Flask App

Peraton

Peraton at a Glance

SPACE | *We protect critical space assets and propel humanity to new worlds.*

INTELLIGENCE | *We provide a secure decision advantage by harnessing emerging, disruptive technologies within a dynamic, digitally pervasive environment.*

CYBER | *We outsmart digital attacks on our nation's most critical networks and systems, and take the fight to the invisible threat.*

DEFENSE | *We're pioneering new frontiers in unmanned vehicles, avionics, mission and enterprise IT and advanced threat detection.*

HOMELAND SECURITY | *We provide critical technology integration, mission support services and lighter than air systems to ensure citizen security and border protection.*

HEALTH | We empower civilian and military health organizations to become more customer-centered, data-driven, efficient and secure through the capture, management and integration of information.

CIVILIAN, STATE & LOCAL | *We improve outcomes and create positive citizen engagement by delivering the right infrastructure, business solutions, and digital transformation services to help achieve policy objectives and integrate citizen-centric services.*

Peraton

Peraton Culture

Teamwork, innovation, and personal growth

- Focus on innovation and staying current with leading-edge technologies
- Team-oriented culture the promotes collaboration and successful outcomes for our customers
- Highly technical, highly capable workforce to learn from
- College graduates
  - Grow your career while making an impact on society
  - Excellent benefits program
- Internships
  - Work on customer-facing projects (no "intern" projects)
  - Prepare for your career – we hire about half of the students who intern or co-op with us

Peraton

Ways to Engage with Peraton

- Stop by our Sponsor table this weekend to meet with a Recruiter

- Apply online at https://careers.peraton.com for future opportunities with Peraton
  - Entry-Level Positions as a Software Engineer, Systems Engineer, Data Scientist, or Cybersecurity Engineer
  - Internship Opportunities

- US citizenship required for all positions

Peraton

Flask Introductory Overview

“Prerequisites”

- General knowledge of Python
- Reminder: how to run a Python app
  - Python <app>.py
  - Py <app>.py
- Familiarity with JSON
- Familiarity with HTTP

Peraton

What is Flask?

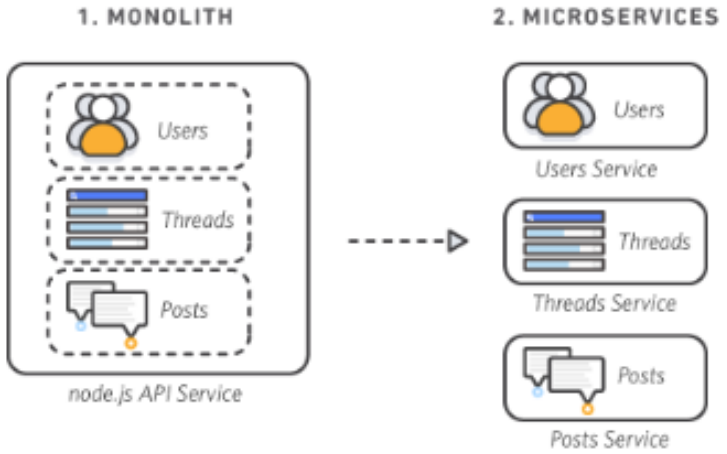
- Lightweight Web Application Microframework
  - Integrates Well with Applications
  - Simple Core
  - Written in Python
  - Relatively Quick Implementation
  - Design Decisions Made By Developer

- What Flask Is Not
  - Designed for large scale applications
  - Asynchronous servers

Flask is designed to facilitate rapid development of web applications and microservices

When to Use Flask

- Web apps
  - Developing a simple web application in a short amount of time
  - Developing a web application without being locked into many dependencies
- Microservices
  - Rapid deployment of RESTful APIs
  - Modular and scalable architecture
    - Separating processes into smaller, separate components
    - Only for an individual task or goal
  - Can be deployed into cloud environments, such as AWS
    - Use with container orchestration service (ECS, EKS, etc.)
    - This creates scalable cloud services



Breaking a monolithic application into microservices

Peraton

Features of Flask

- REST APIs/Request Object
  - Can build REST APIs
- Error Handling
  - Can register error handlers to show custom error pages
- Templates
  - Jinja2 template engine
  - Render_template() method

Flask provides a foundation on which a web application can be built

Technologies Flask Leverages

- Jinja
  - Template language
- WSGI/Werkzeug
  - Werkzeug implements WSGI
  - Python interface between applications and servers
- Additional Dependencies
  - MarkupSafe
  - ItsDangerous
  - Click
- Optional Dependencies
  - Blinker
  - Python-dotenv
  - Watchdog

Flask Deep Dive

Rest APIs

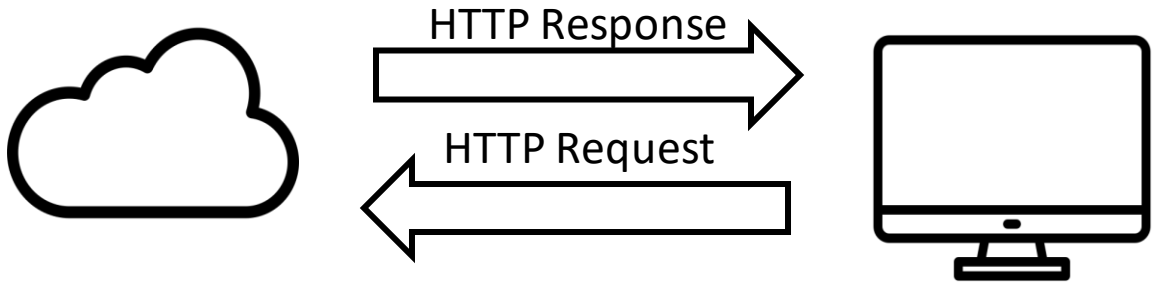
- REpresentational State Transfer (REST)
  - Remote services that provide functionality
  - Typically use CRUD operations
  - Stateless
- Endpoints added onto IP or URIs
  - Determines how the service handles the request
  - Example:
    - `127.0.0.1:8000/hello-world/`
    - `<IP Address>:<Port Number>/<endpoint>/`
- Flask can be used to create RESTful APIs
  - `@app.route('/<endpoint>/')` adds an endpoint
  - app is declared by assigning it to flask
  - `app.run()` starts flask

```python
from flask import Flask

app = Flask(__name__)

@app.route('/hello-world/')
def hello_world():
    return 'Hello World'

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



Peraton

Request Object

- A python object containing data transferred
- Provides access to:
  - Request Args
    - Can be added onto URIs to provide information
  - Input Data or Files
  - User data (Cookies)
  - Request Headers
- Important Functions or Parameters
  - `get_json()` returns json sent by the user
  - `get_data()` returns any data sent by the user
  - `args.get(<Argument>)` gets arguments from the url

```
@app.route('/', methods=['GET'])
def hello_world():
    return 'Hello World'

@app.route('/birthday/', methods=['POST'])
def birthday():
    input_json = request.get_json()
    birthday = input_json['birthday']
    first = input_json['first']
    last = input_json['last']
    return first + ' ' + last + "'s birthday is " + birthday

@app.route('/', methods=['POST'])
def invert():
    req = request.get_data()
    mime = request.content_type
    pil_image = ImageOps.invert(Image.open(io.BytesIO(req)))
    return send_file(convert_PIL_image(pil_image), mimetype=mime)
```

Error Handling

- Flask allows developer to handle errors and provide feedback in multiple ways
  - `@app.errorhandler` tag
    - Sets the response for all errors of the given code
    - Can return HTML or JSON for specific error codes
  - `abort(HTTP_Status_Code)`
    - Stops the request with the given HTTP error
  - `flask.Response(Text, HTTP_Status_Code)`
    - Returns a specific error
- Logging can be done using `app.logger` to log events
  - `app.logger.debug()`
  - `app.logger.warning()`
  - `app.logger.error()`

Code	Meaning
200	Ok
400	Bad Request
401	Unauthorized
404	Not Found
500	Internal Service Error

Sample Error Code

```
@app.route('/fourzerozero/')
def user_profile():
    abort(400) # Returns a 400 error with 'bad request!' as text

@app.route('/fourzerofour/')
def fourzerofour():
    abort(404) # Returns standard 404 error after ending the request

@app.route('/twozerotwo/')
def twozerotwo():
    return flask.Response(response='Here is an example of a 202', status=202)

@app.errorhandler(werkzeug.exceptions.BadRequest)
def handle_bad_request(e):
    return 'bad request!', 400
```

Templates

- Flask leverages the Python template engine, Jinja2
  - Generates HTML but is not limited to HTML
  - Many other formats
    - Markdown, Plaintext, XML, CSS, and plenty more.
- render_template('fileneame.html')
  - Will look in the templates folder
    - In a module it will be next to the python file
    - In a package it will be inside the package
- Similar to HTML but adds functionality
  - Control structures
  - Math
  - Comparisons and Logic
  - Code Injection
  - Automatically Escape Characters
  - Can bind data to webpages

Python (Flask)

```
from flask import render_template

todo_list = ['Eat', 'Sleep', 'Code']

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

HTML Template:

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

<h2>Todo List</h2>
<ul>
        {% for item in todo_list %}
            <li>{{item}}</li>
        {% endfor %}
</ul>
```

Web Framework Examples

- Bootstrap
  - CSS framework for front-end web development
  - Builds quick and responsive websites
  - Great with templates
  - Bootstrap · The most popular HTML, CSS, and JS library in the world. (getbootstrap.com)

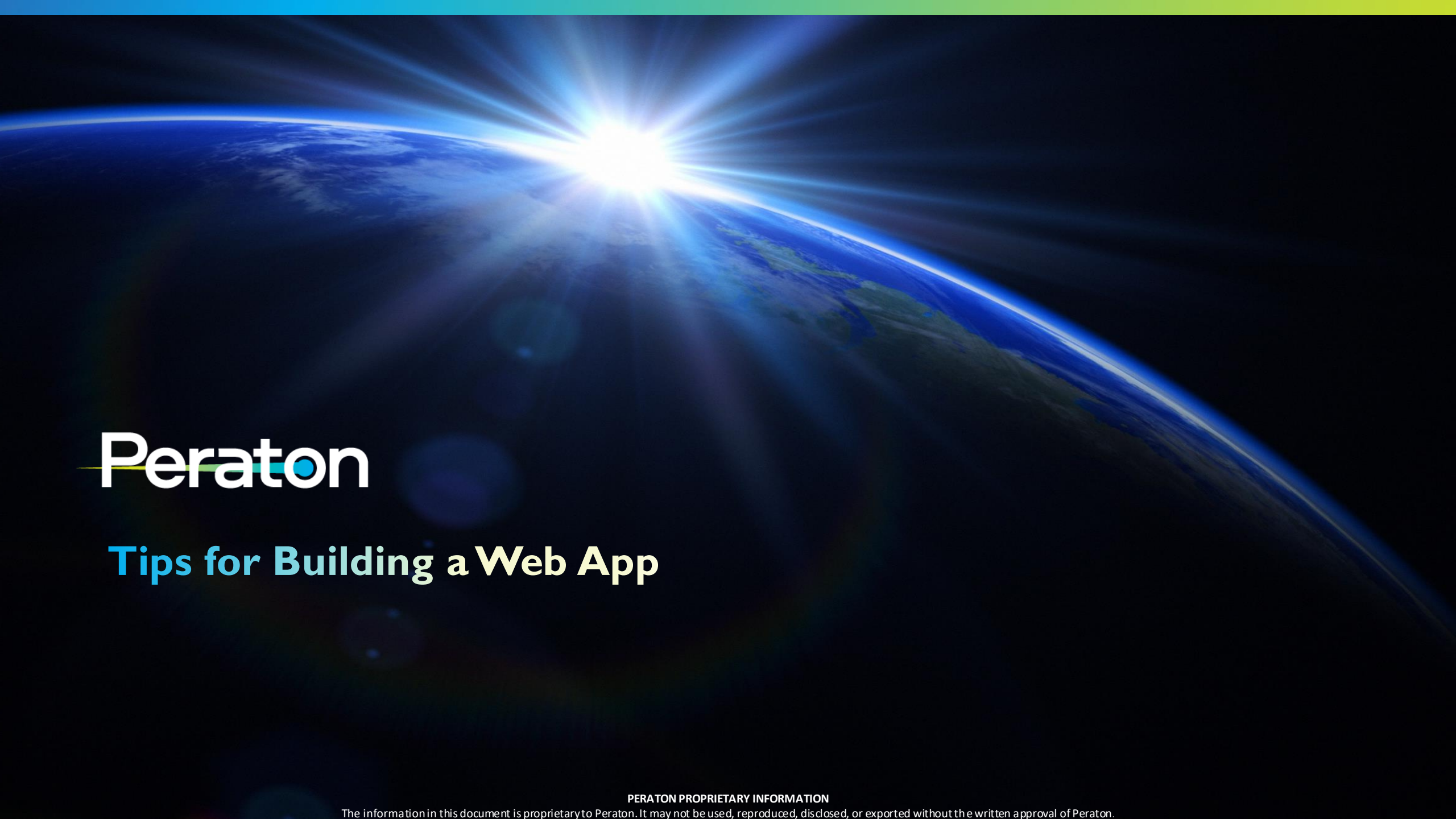
- React
  - Structured web applications with a focus on user interface design
  - JavaScript library developed by Meta
  - React – A JavaScript library for building user interfaces (reactjs.org)

- Angular
  - Structured, component-based, single-page applications
  - TypeScript framework developed by Google
  - Angular

Extensions

- Flask-bootstrap
  - Makes using Bootstrap CSS framework easier
- Flask Debug-toolbar
  - Debugging sidebar
- Flask-Security-Too
  - Facilitates implementation of security features
- Find Flask Extensions
  - PyPI packages tagged "Framework :: Flask"

Common Naming Conventions for Flask Extensions: "Flask-Name" or "Name-Flask"



Tips for Building a Web App

Recommended Process

- Think about what you want the web app to do, set initial requirements/goals
  - Consider the users
  - Determine any constraints
  - Determine goal of the web application
- Brainstorm ideas
- Create Initial Wireframe
  - By hand or use wireframing tools
- Focus on creating a Minimum Viable Product (MVP)
  - Once MVP is reached, add enhancements

Get Started with Flask: Installation

- Note: Flask requires Python 3.7 or newer
- Follow installation instructions
  - Installation — Flask Documentation (2.2.x) (palletsprojects.com)
- Run through Flask Tutorial (if desired)
  - Tutorial — Flask Documentation (2.2.x) (palletsprojects.com)
- Helpful Development Tools and related technologies
  - Good code editors
    - Visual Studio Code
    - PyCharm: the Python IDE
    - Notepad++
  - HTTP request tool
    - Postman API Platform
  - Web Development Tools
    - Press F12 in Firefox, Chrome, or Edge

Demo Sample Application

BenWynnVT/**FlaskSampleCode (github.com)**