FLASK APIS

TATRAS
OPTIMIZE | INNOVATE | DISRUPT

SABUDH

1. Overview
2. Methods
3. Payloads - Request/Response
4. HTTP Codes
5. Blueprints

WHAT IS FLASK. WHY USE IT?

Flask is a web development framework developed in Python. It is easy to learn and use. Flask is “beginner-friendly” because it does not have boilerplate code or dependencies, which can distract from the primary function of an application.

Advantages of Flask

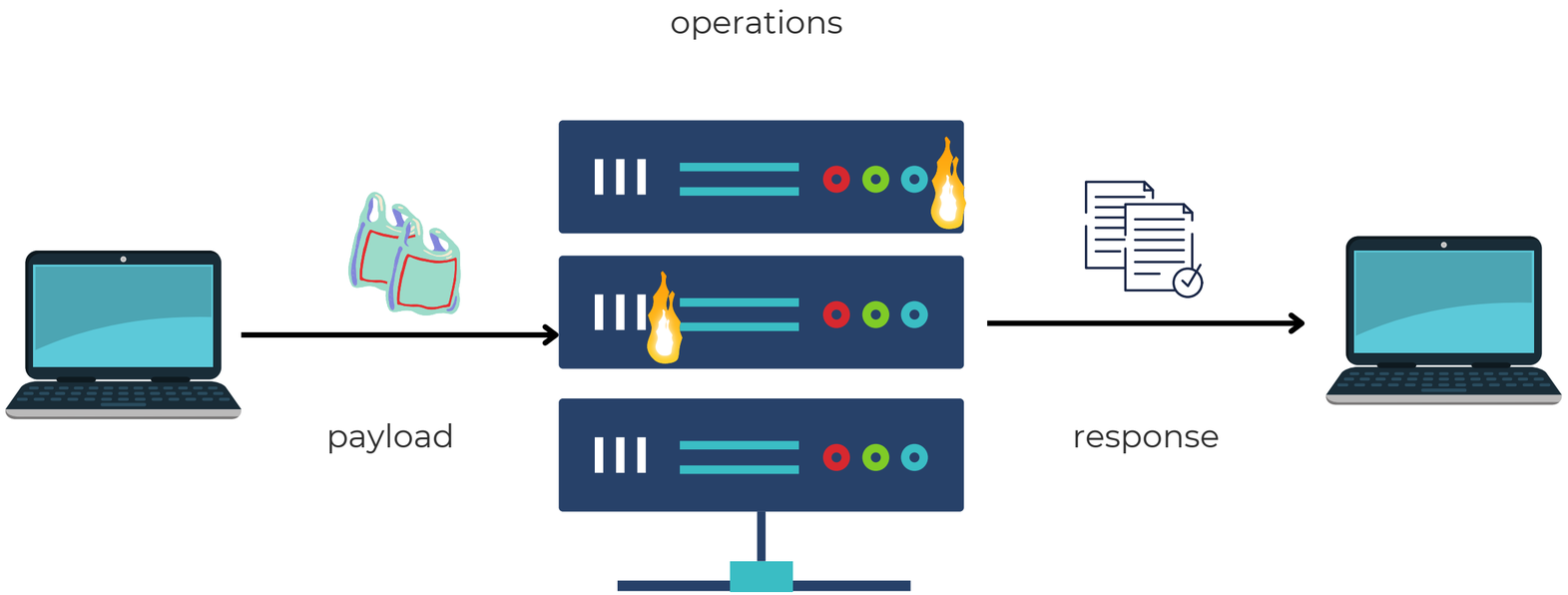
- Scalable
  - Size is everything, and Flask’s status as a microframework means that you can use it to grow a tech project such as a web app incredibly quickly.
- Flexible
  - This is the core feature of Flask, and one of its biggest advantages. To paraphrase one of the principles of the Zen of Python, simplicity is better than complexity, because it can be easily rearranged and moved around.
- Lightweight
  - There are few constituent parts that need to be assembled and reassembled, and it doesn’t rely on a large number of extensions to function. Each module acts as an independent building block, which can execute one part of the functionality.
- Documentation
  - Following the creator’s own theory that “nice documentation design makes you actually write documentation,” Flask users will find a healthy number of examples and tips arranged in a structured manner.

HOW API'S WOKR

1. Request
   A request payload which can be json, xml, params.

2. Operation
   these can include db operations, predictions, changes in data

3. Response
   A response which usually is json, xml.



METHODS

Commonly used methods.

GET

THE GET METHOD REQUESTS A REPRESENTATION OF THE SPECIFIED RESOURCE. REQUESTS USING GET SHOULD ONLY RETRIEVE DATA.

CRUD - READ

POST

THE POST METHOD SUBMITS AN ENTITY TO THE SPECIFIED RESOURCE, OFTEN CAUSING A CHANGE IN STATE OR SIDE EFFECTS ON THE SERVER.

CRUD - CREATE

PATCH

THE PATCH METHOD APPLIES PARTIAL MODIFICATIONS TO A RESOURCE.

CRUD - UPDATE

DELETE

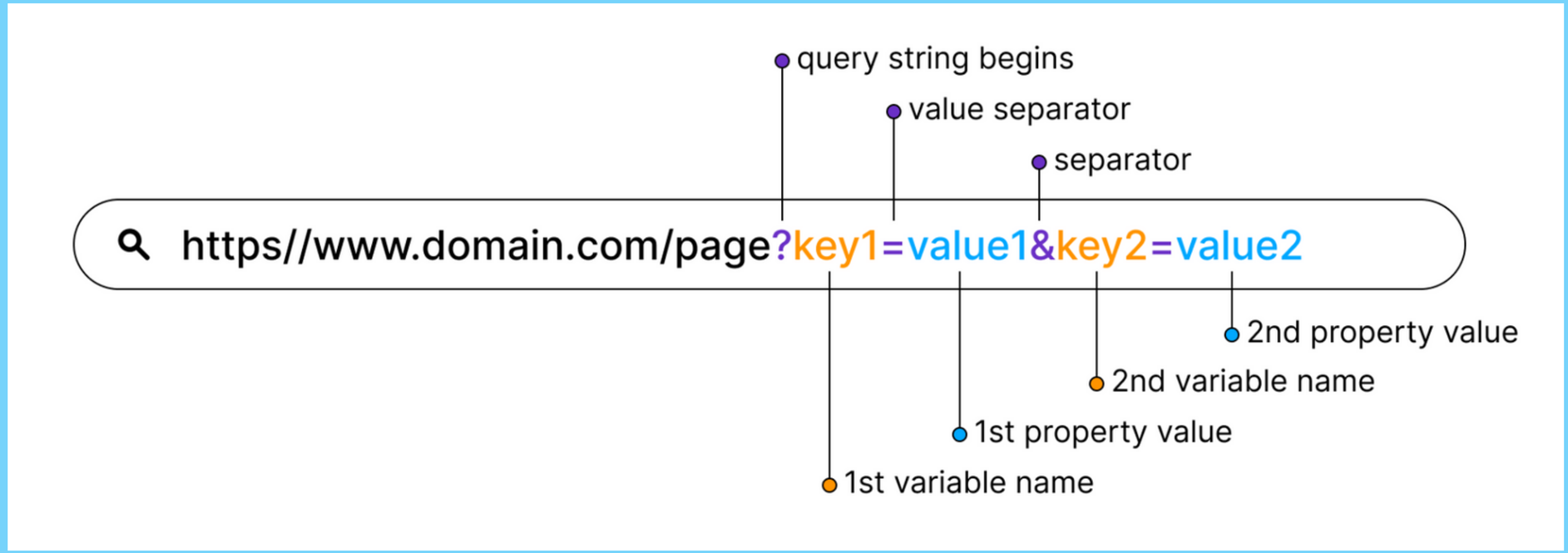
THE DELETE METHOD DELETES THE SPECIFIED RESOURCE.

CRUD - DELETE

OTHER METHODS - OPTIONS, TRACE, HEAD, CONNECT, PUT

PARAMS

REQUEST PARAMETERS ARE USED TO SEND ADDITIONAL INFORMATION TO THE SERVER



FORM DATA

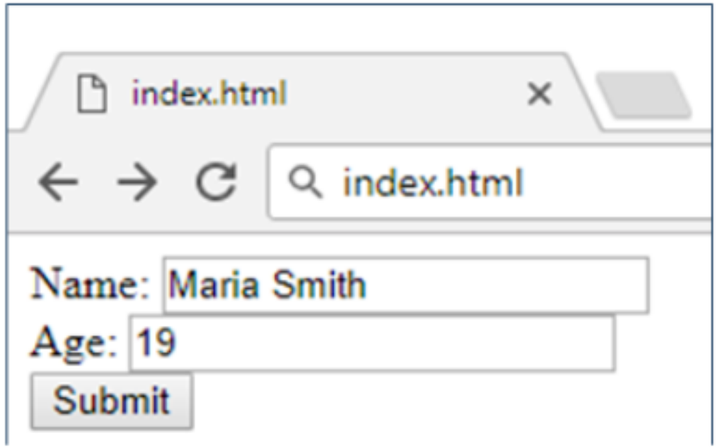
FORM DATA OBJECT LETS YOU COMPILE A SET OF KEY/VALUE PAIRS TO SEND.

```
<form method="post">
Name: <input type="text" name="name"/> <br/>
Age: <input type="text" name="age"/> <br/>
<input type="submit" />
</form>

POST /index.html HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 23

name=Maria+Smith&age=19
```

URL-encoded form data



JSON

JSON STANDS FOR JAVASCRIPT OBJECT NOTATION. IT IS A LIGHTWEIGHT FORMAT FOR STORING AND TRANSPORTING DATA, OFTEN USED WHEN DATA IS SENT FROM A SERVER TO A WEB PAGE.

```
JSON Object →  {                                  String Value
                 "company": "mycompany",
                 "companycontacts": {  ←  Object Inside Object
                   "phone": "123-123-1234",
                   "email": "myemail@domain.com"
                 },
                 "employees": [  ←  JSON Array
                   {
                     "id": 101,
                     "name": "John",
Array Inside Array →  "contacts": [
                       "email1@employee1.com",
                       "email2@employee1.com"
                     ]
                   },
                   {
                     "id": 102,  ←  Number Value
                     "name": "William",
                     "contacts": null  ←  Null Value
                   }
                 ]
               }
```

OTHER REQUEST TYPES:

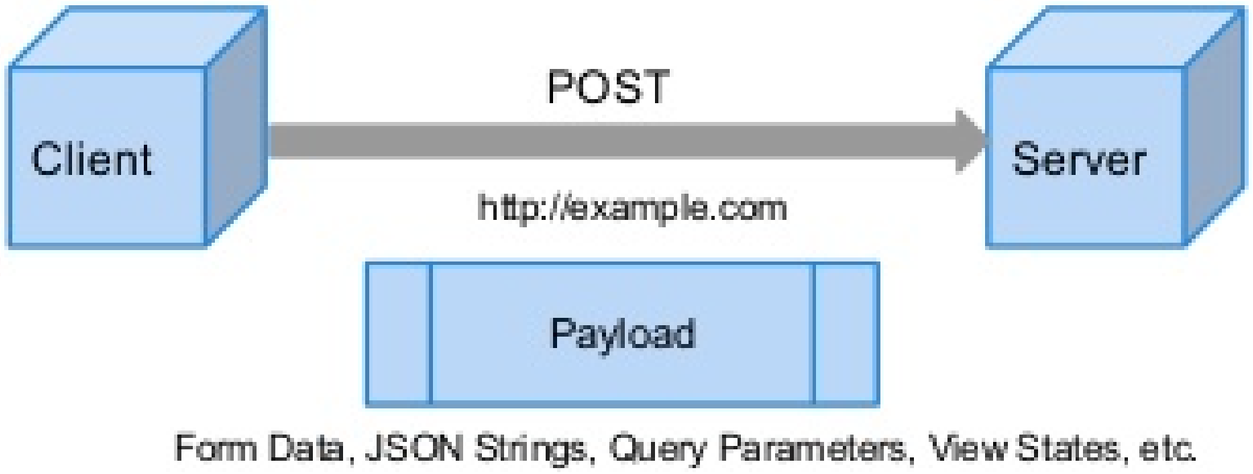
- HTML
- XML
- JAVASCRIPT

GET METHOD

Requests using GET should only retrieve data.

POST METHOD

The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server.



RESPONSE PAYLOAD

The following example shows the JSON structure received in response to a request

```
{
  "data": str....int....dict....array...,
  "message": bool
}
```

Status Code: 2xx,4xx....

HTTP CODES

1xx: Information

1. 100: Continue

2xx: Success

1. 200: OK
2. 201: Created
3. 202: Accepted
4. 204: No Content

3xx: Redirect

1. 301: Moved Permanently
2. 307: Temporary Redirect

4xx: Client Error

1. 400: Bad Request
2. 401: Unauthorized
3. 403: Forbidden
4. 404: Not found

5xx Server Error

1. 500: Internal Server Error
2. 502: Bad Gateway
3. 503: Service Unavailable
4. 504: Gateway Timeout

BLUEPRINT

Each Flask Blueprint is an object that works very similarly to a Flask application. They both can have resources, such as static files, templates, and views that are associated with routes.
However, a Flask Blueprint is not actually an application. It needs to be registered in an application before you can run it. When you register a Flask Blueprint in an application, you're actually extending the application with the contents of the Blueprint.

Organizing Your Projects
Instead of structuring the application inside one file, you can leverage a Flask Blueprint to split the code into different modules

Directory Structure

```
root/
|
├── blueprints/
|   └── model_api.py
|
├── modules/
|   └── model.py
── app.py
── requirements.txt
── model.pkl
── vector.pkl
```

THINGS TO TRY

- Create APIs using UPDATE, DELETE methods.
- Create api to train a model.
- Play around with different request types.

Questions?

Pulkit Komal
Email - pulkit.k@tatrasdata.com

Thank You