Server-Side Web Programming: CGI (Part 2)

Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover…
  - CGI programming review
  - The HTTP GET method vs. the HTTP POST method

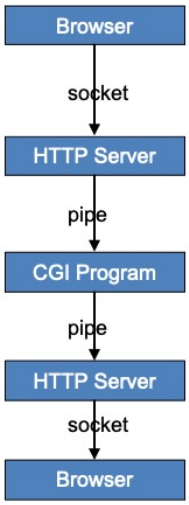
Agenda

- **CGI Review**
- GET vs. POST

CGI Review

- **Common Gateway Interface (CGI)**
  - The oldest and simplest way to implement a Web app that generates content dynamically

CGI Review



HTTP request that specifies a program and name=value pairs
GET: name=value pairs in request header
POST: name=value pairs in request body

Name=value pairs
GET: name=value pairs in QUERY_STRING env var
POST: name=value pairs in stdin

HTTP response (partial headers & complete body)
Writes to stdout

HTTP response (complete headers & complete body)

CGI Review: GET vs. POST

- To command browser to issue a HTTP **GET** request:
  - (1) Enter URL at top of browser
  - (2) Click on page link
  - (3) Submit a form whose method="get"
- To command browser to issue a HTTP **POST** request:
  - (1) Submit a form whose method="post"

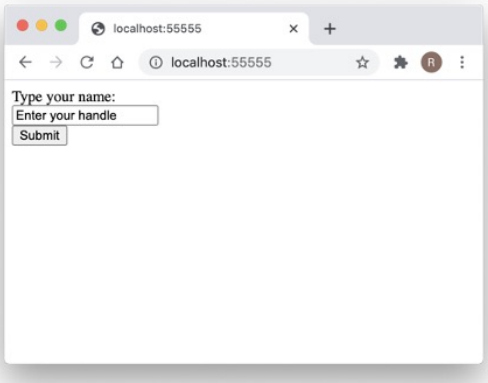
CGI Review: GET Method

- See **HelloPythonGet** app

```
$ python runserver.py 55555
Serving HTTP on 0.0.0.0 port 55555 (http://0.0.0.0:55555/) ...
```

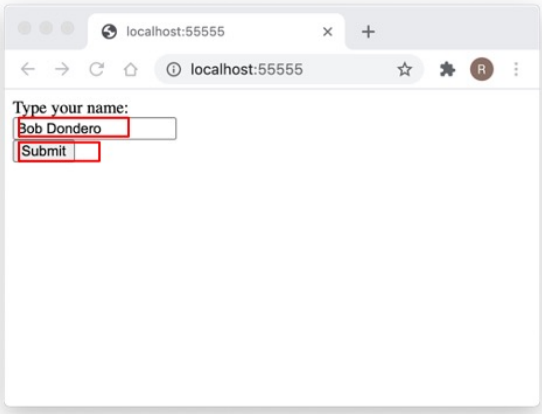
CGI Review: GET Method

- See **HelloPythonGet** app



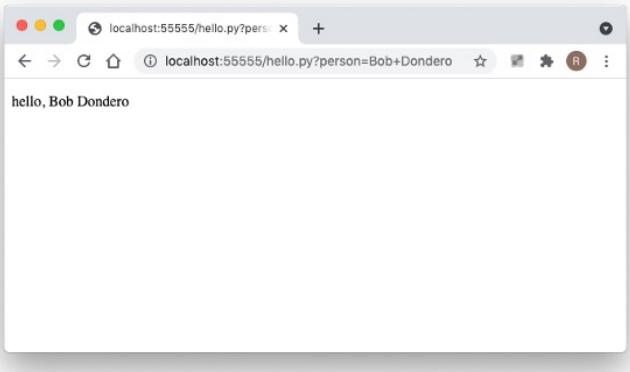
CGI Review: GET Method

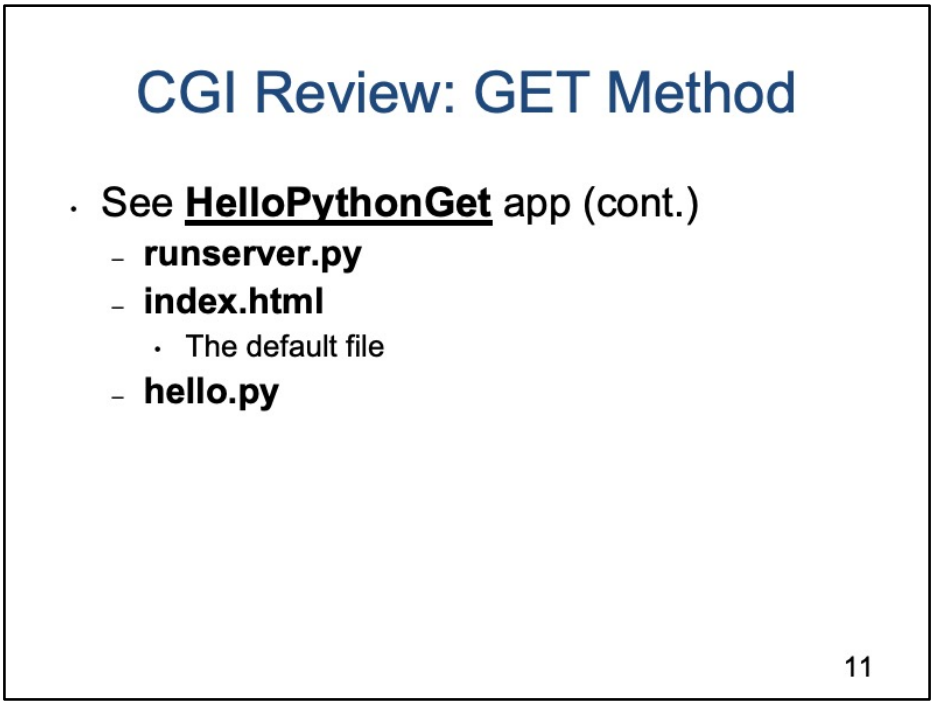
- See **HelloPythonGet** app



CGI Review: GET Method

- See **HelloPythonGet** app





CGI Using GET

[see slide]

Code notes: runserver.py

Launches simplehttpserver.py with the --cgi argument

When simplehttpserver.py sees a request for a .py file…

It interprets that request as a CGI request

I'll create a runserver.py file for every web app

So I (and you) need not remember how to launch the HTTP server

Code notes: index.html

simplehttpserver.py considers index.html to be the default file

If HTTP request doesn't name a file, the file defaults to index.html

Defines a form

Submitting the form will send a HTTP GET request for hello.py with name=value pair person=[whatever the user enters]

Code notes: hello.py

Uses Python environ variable (a dict object) to get the value of the QUERY_STRING environment variable

Fetches name/value pairs from QUERY_STRING environment variable
Composes HTTP response
Writes it to stdout

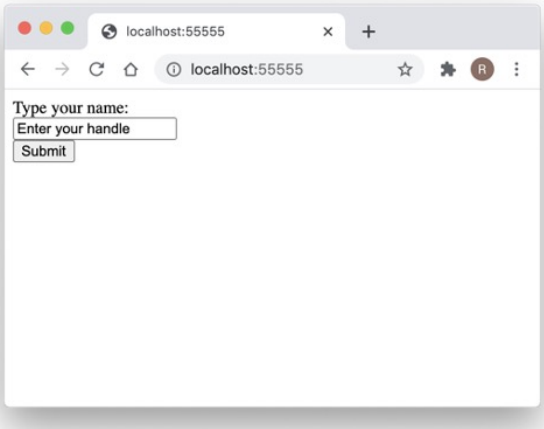
CGI Review: POST Method

- See **HelloPythonPost** app

```
$ python runserver.py 55555
Serving HTTP on 0.0.0.0 port 55555 (http://0.0.0.0:55555/) ...
```

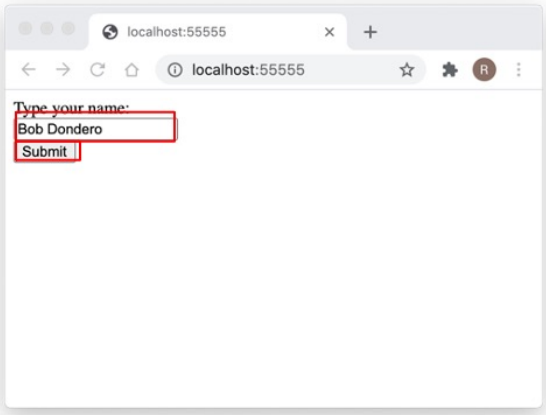
CGI Review: POST Method

- See **HelloPythonPost** app



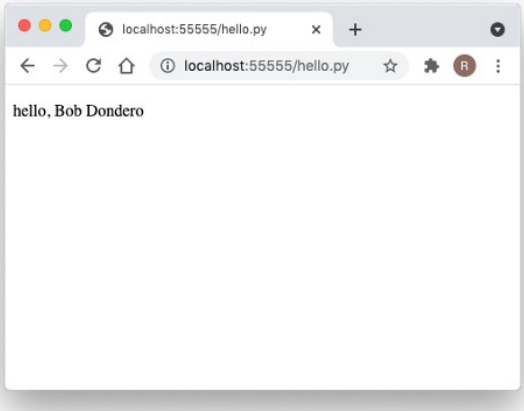
CGI Review: POST Method

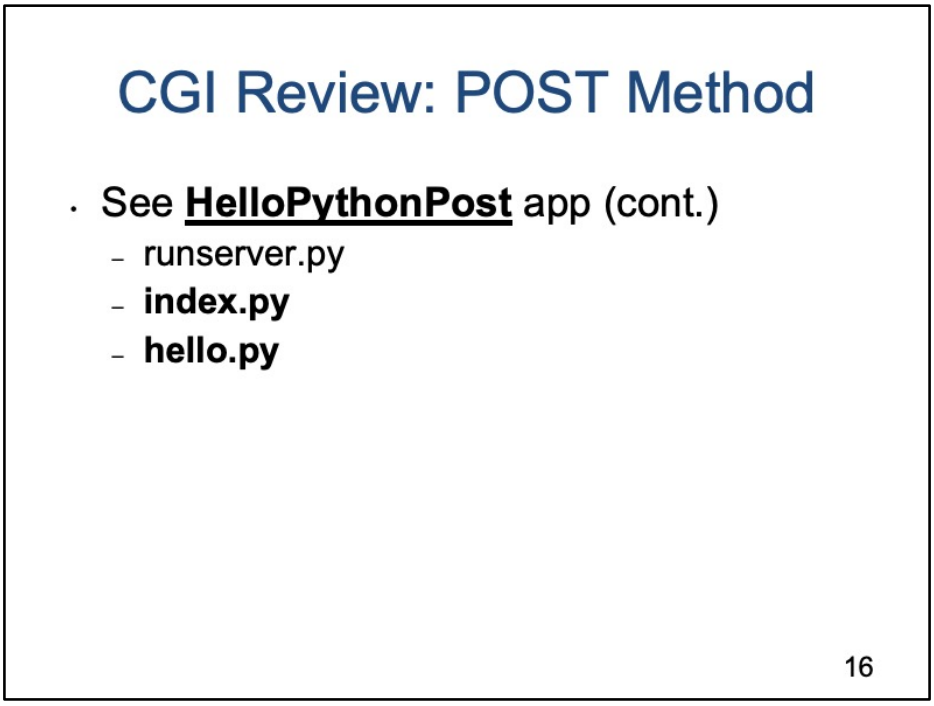
- See **HelloPythonPost** app



CGI Review: POST Method

- See **HelloPythonPost** app



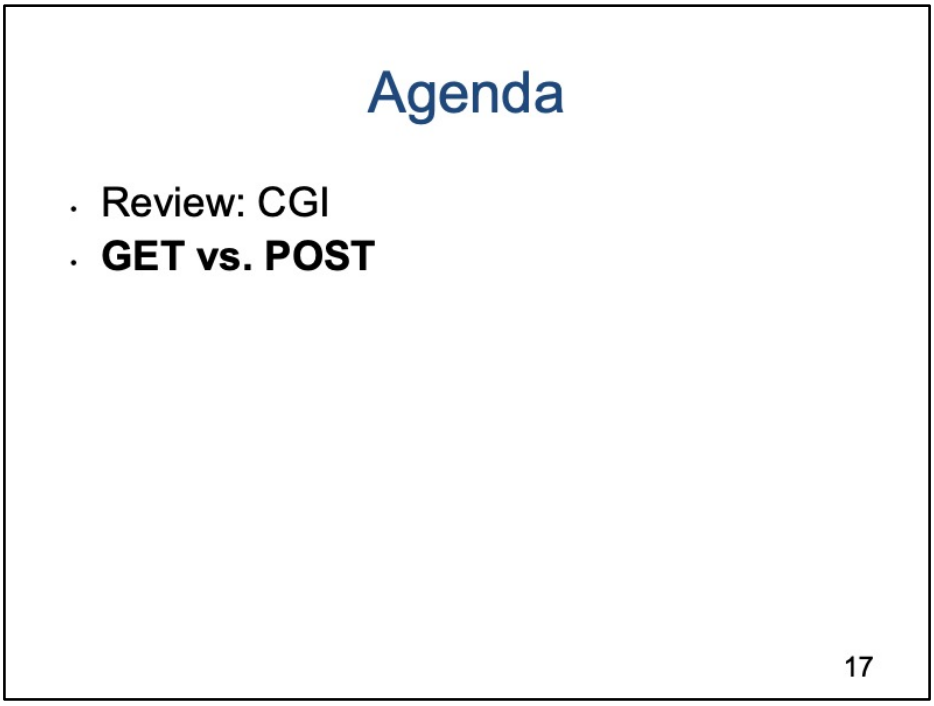


CGI Using POST

[see slide]

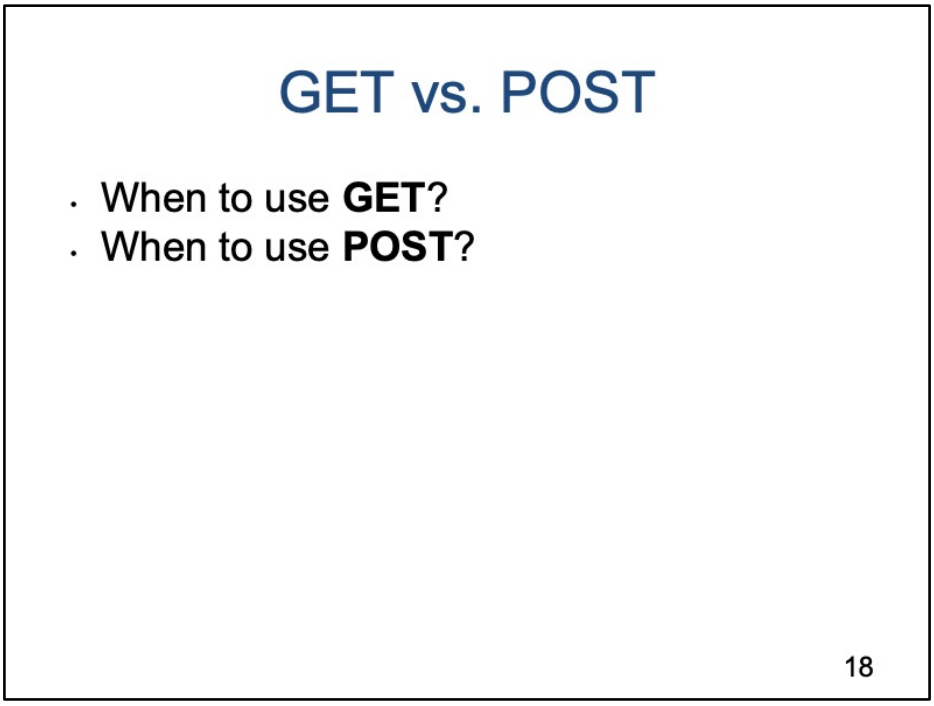
Code notes: index.py
Form will submit a POST request

Code notes: hello.py
Fetches name=value pairs from its stdin, not from an env var



GET vs POST

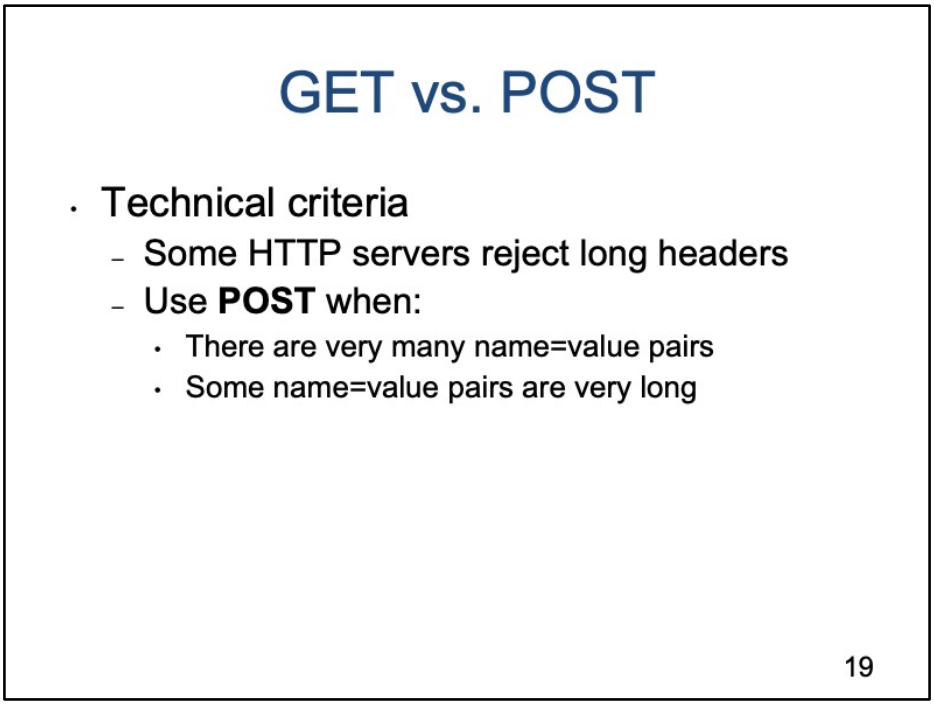
Let's again sharply contrast the two HTTP methods…
And also cover when you should one vs. the other



GET vs. POST

[see slide]

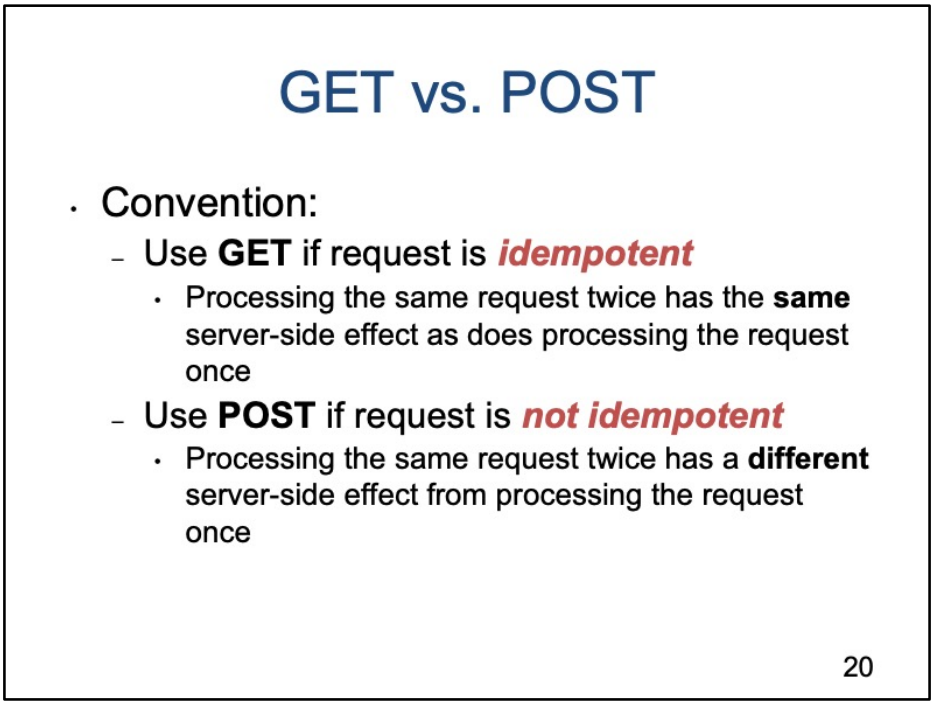
The GET and POST methods have the same power
When should you use GET? POST?



GET vs. POST

Technical criteria

The HTTP spec defines no header size limits, but…
Some HTTP servers reject long headers
Apache: 8K limit
Microsoft IIS: 16K limit
If there are very many name=value pairs, or some name=value pairs are very long, then
They might not fit in the request header
You might not be able to use GET
You might need to use POST



GET vs. POST

Oxford dictionary:

> **Idempotent**: denoting an element of a set which is unchanged in value when multiplied or otherwise operated on by itself

[see slide]

GET vs. POST

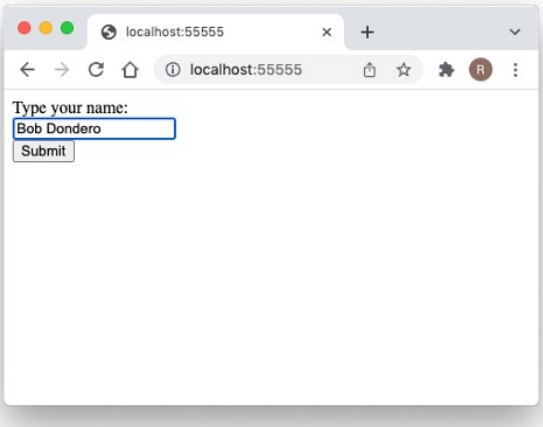
- Convention:
  - **Example**: HTTP request asks for car price
    - Processing the request twice = processing the request once
    - Request is **idempotent**
    - Request should use **GET**
  - **Example**: HTTP request purchases a car
    - Processing the request twice != processing the request once
    - Request is **not idempotent**
    - Request should use **POST**

GET vs. POST

- With that convention...
- Browser has sent **GET** request =>
  - Browser assumes request was **idempotent**
  - Browser **does not warn** about page refresh
- Browser has sent **POST** request =>
  - Browser assumes request was **not idempotent**
  - Browser **warns** about page refresh

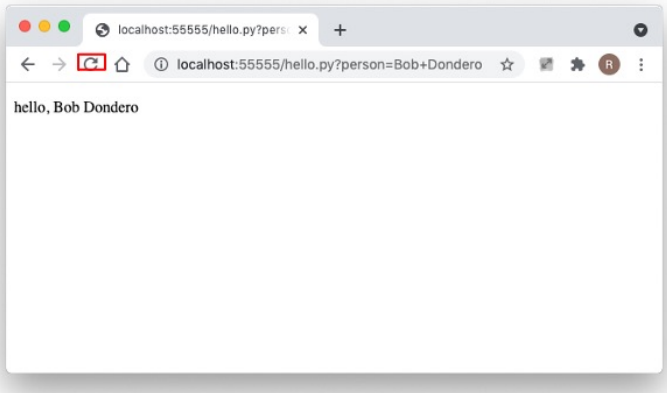
GET vs. POST

- **Example**: HelloPythonGet



GET vs. POST

- **Example**: HelloPythonGet (cont.)



Reload this page

24

GET vs POST

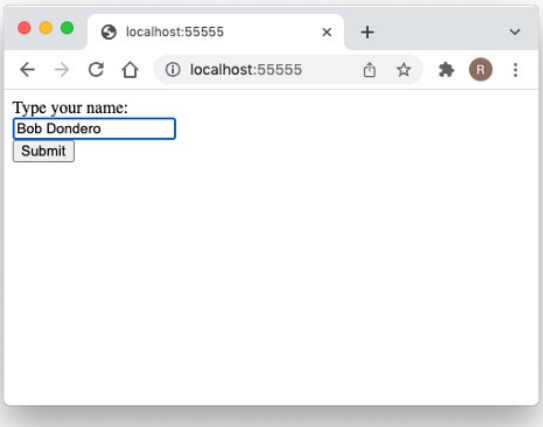
The HTTP request that yielded the current page was a GET request
So the browser assumes the request was idempotent

GET vs POST

So a page reload (causing the browser to send the same HTTP request) generates no browser warning

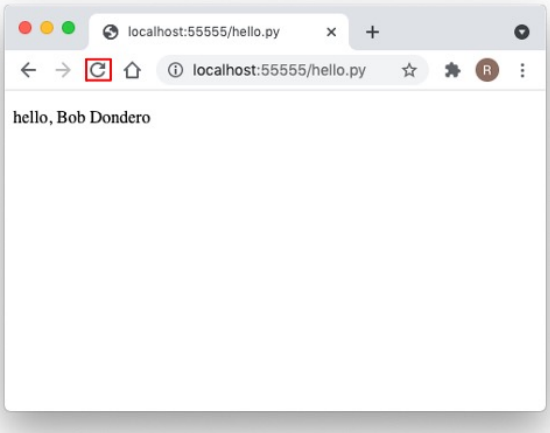
GET vs. POST

- **Example**: HelloPythonPost



GET vs. POST

- **Example**: HelloPythonPost (cont.)



Reload this page

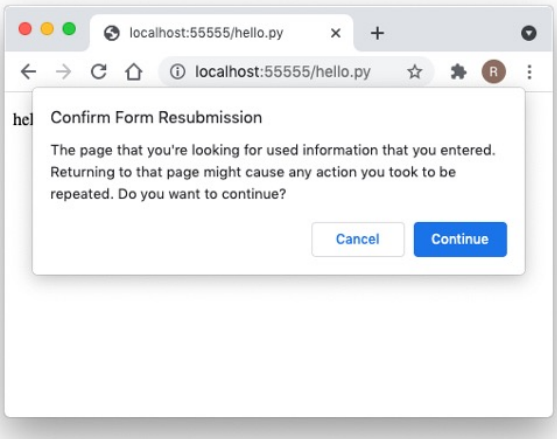
27

GET vs POST

The HTTP request that yielded the current page was a POST request
So the browser assumes the request was not idempotent

GET vs. POST

- **Example**: HelloPythonPost (cont.)



Warning!

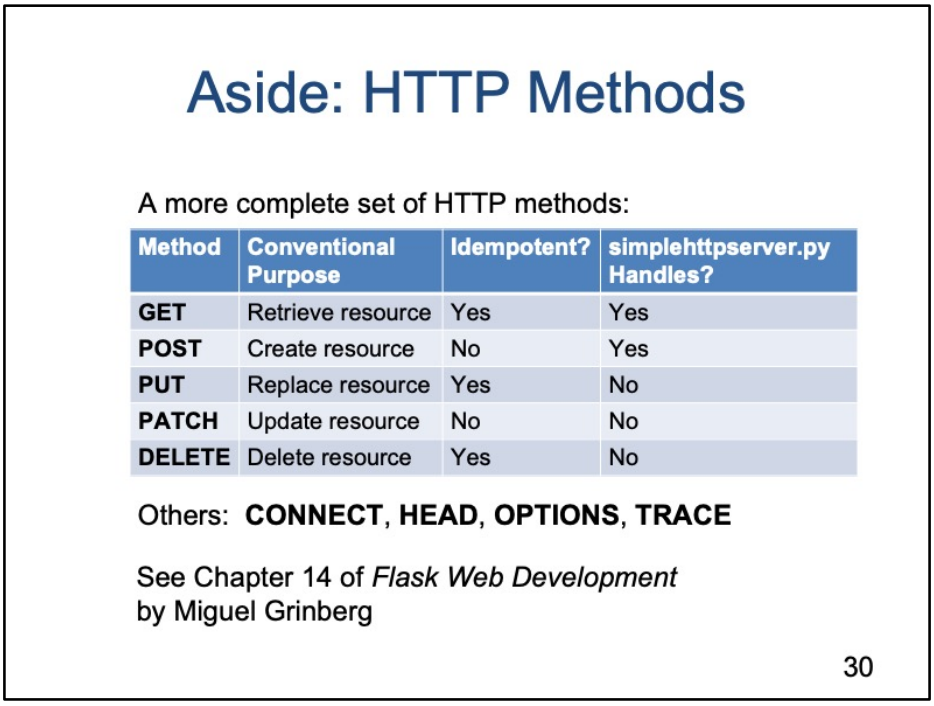
28

GET vs POST

So a page reload (causing the browser to send the same HTTP request) generates a browser warning

GET vs. POST

- Generally:
  - Use **GET** to **retrieve** server-side state
  - Use **POST** to **change** server-side state



Aside: HTTP Methods

[see slide]

An application that uses HTTP, and uses the HTTP methods for their conventional purposes, is said to be RESTful

Summary

- We have covered:
  - CGI programming review
  - GET vs. POST