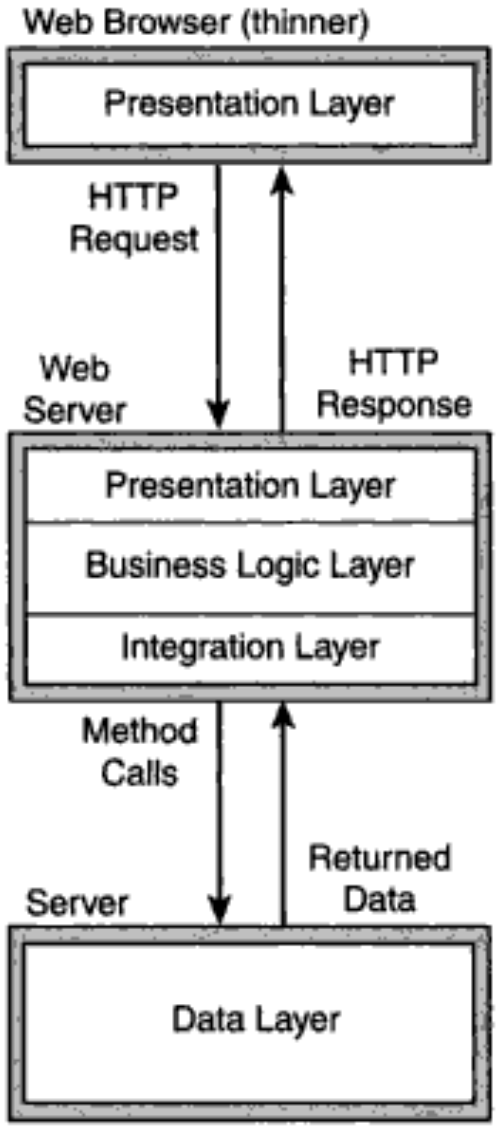
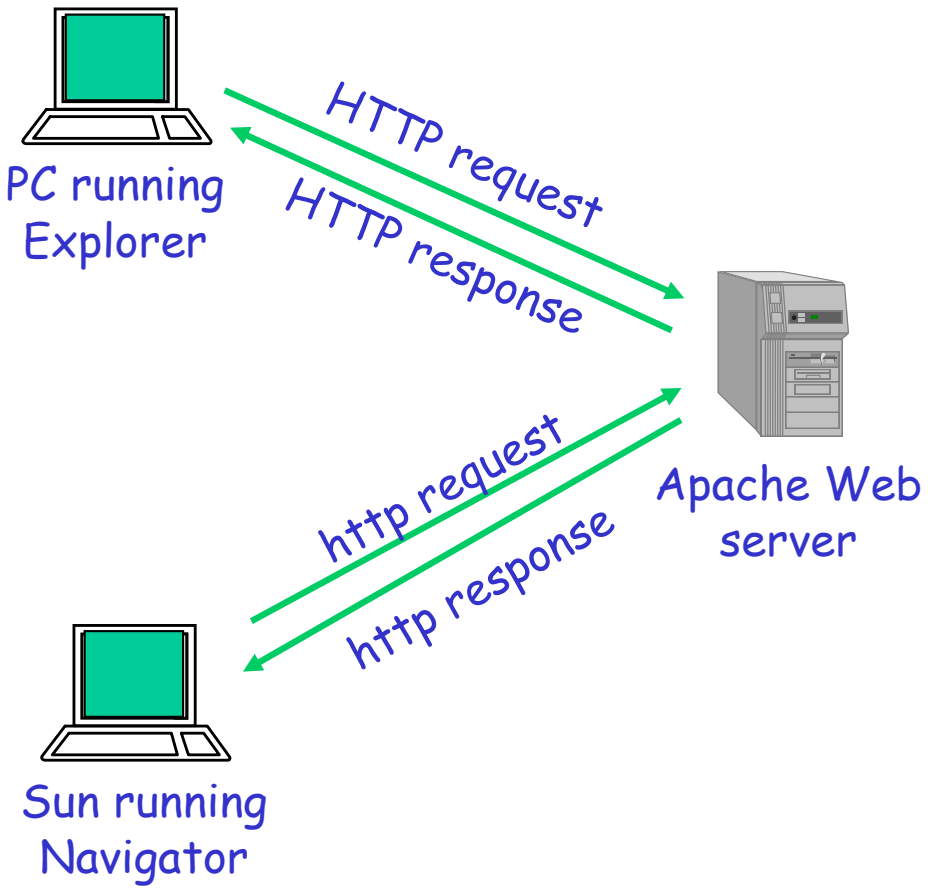
Web Application Development
Server Side

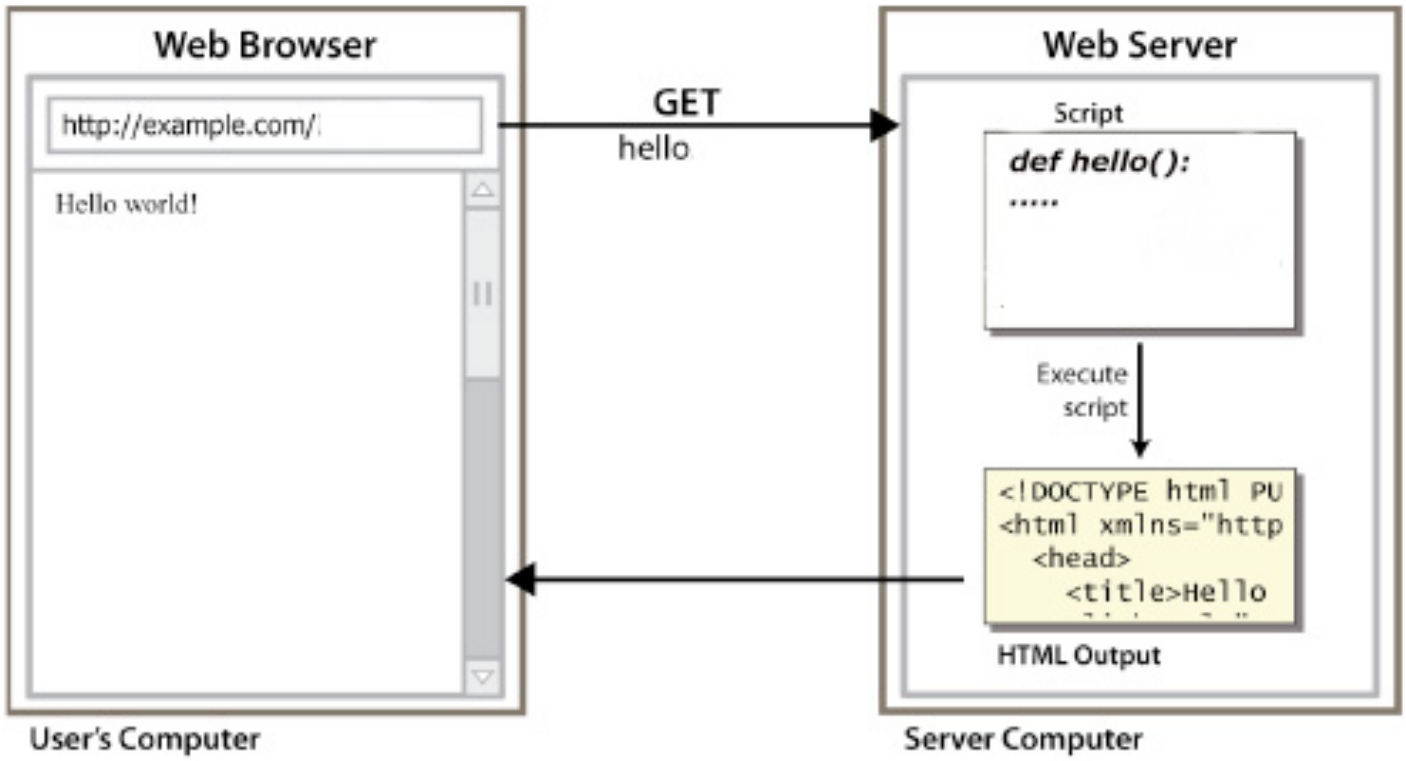
Architecture



Client-Server Paradigm



Request Life Cycle



Flask is a microframework for Python based on Werkzeug, Jinja 2 and good intentions.

http://flask.pocoo.org/

“Easy” to learn

Plug-in architecture provides rich functionality

Hello World

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'

app.run()
```

Client-Handler Interaction

- Clients request the execution of a handler program by means of a:
  - A request method (e.g., GET, POST)
  - Universal Resource Identifier (URI)
    - Identifies handler
  - Extra arguments

URI Syntax

<protocol>://<host><port>/<path-info><script>"?"<query-string>

http://finance.yahoo.com/add?op1=10&op2=20

<protocol>	http
<host>	finance.yahoo.com
<port>	80
<path-info>	null
<script>	add
<query-string>	op1=10, op2=20

RFC 2396 Encoding

- All arguments
  - Single string of ampersand (&) separated name=value pairs

    name_1=value_1&name_2=value_2&...

- Spaces
  - Replaced by a plus (+) sign

- Other characters (ie, =, &, +)
  - Replaced by a percent sign (%) followed by the two-digit hexadecimal ASCII equivalent

Method

- GET
  - Arguments appear in the URL after the ?
  - Can be expressed as a URL
  - Limited amount of information can be passed this way
  - URL may have a length restriction on the server
  - Arguments appear on server log

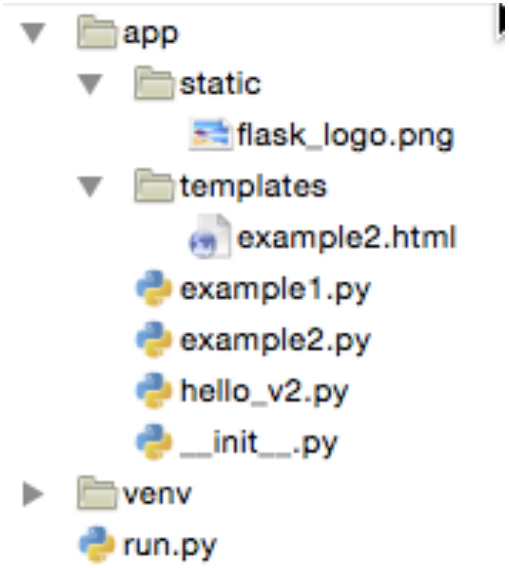
- POST
  - Arguments are sent in the HTTP message body
  - Cannot be expressed as URL
  - Arbitrarily long form data can be communicated (some browsers may have limits (i.e. 7KB)).
  - Arguments usually does not appear in server logs.

Flask request object

- Provides access to http request arguments and environment attributes

form	parsed form data from POST or PUT requests
args	parsed contents of the query string
values	contents of both form and args
cookies	contents of all cookies transmitted with request
environ	the underlying WSGI environment
method	current request method

Application Structure



__init__py

```
from flask import Flask

webapp = Flask(__name__)

from app import hello_v2
from app import example1
from app import example2
```

run.py

```
#!venv/bin/python
from app import webapp
webapp.run(debug=True)
```

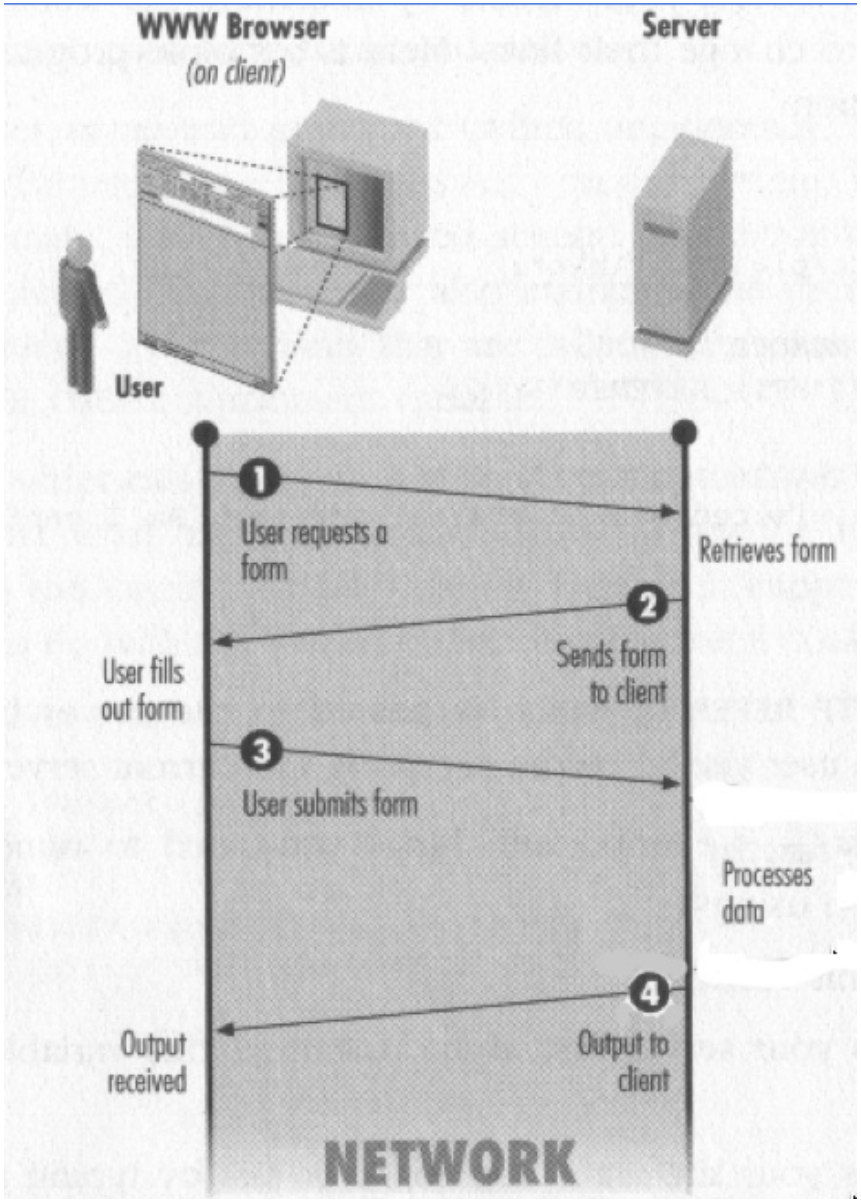
hello_v2.py

```
from app import webapp

@webapp.route('/')
def hello_world():
    return 'Hello, World!'
```

Forms

- Specify request method in "method" attribute
- Automatically encodes all **named** field values



Templates

app/example2.py

```
from flask import render_template, url_for
from app import webapp

import datetime

@webapp.route('/time_and_logo2')
def example2():
    time = datetime.datetime.now()
    imgURL = url_for('static', filename='flask_logo.png')
    return render_template("example2.html",
                           time=time,
                           imgURL = imgURL)
```

app/templates/example2.html

```
<!DOCTYPE html>
<html>
    <img width='100px' src="{{ imgURL }}" />
    Current Date and Time: {{ time }}
</html>
```

Session Maintenance

- HTTP is stateless by default
  - No memory of browsers state
- Motivations for preserving state?
  - Enabling multi-step interactions with users
- Examples:
  - access control (i.e., login)
  - shopping cart
  - web site analytics

Flask Sessions

Data stored on client as encrypted cookie

Initialize session

```
app.secret_key = '\x8\xbeHJ:\x9f\xf0!\xb1a\xaa\x0f\xee'
```

Set session attribute

```
session["loggedIn"] =true
```

Retrieve session attribute

```
count = session["count"]
```

Invalidate session

```
session.clear()
session.pop("count”)
```

Count Visits

app/example5.py

```
from flask import render_template, session
from app import webapp

import datetime

webapp.secret_key = '\x80\xa9s*\x12\xc7x(\x03\xbeHJ:\x9f\xf0!\xb1a\xaa\x0f\xee'

@webapp.route('/count',methods=['GET','POST'])
def count():
    numtimes = 0
    if 'numtimes' in session:
        numtimes = session['numtimes']
    session['numtimes'] = numtimes + 1

    return render_template("example5.html",numtimes=numtimes)
```

app/templates/example5.html

```
<!DOCTYPE html>
<head>
    <title>Count</title>
</head>
<html>
    <body>
        <form method='post' action=''>
            Called {{ numtimes }}
            <input type='submit' value='Count'>

        </form>
    <body>
</html>
```