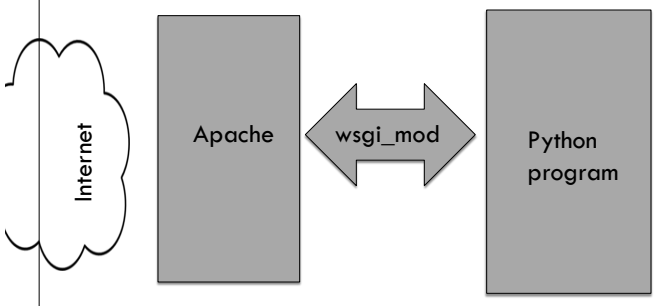
ECS 89

Announcements

- Second program will be up Friday, due Tuesday 4/29 at 10pm
- Does everyone have a pedometer?

Server side - wsgi



Apache

- Is always running (for us, on pc110), ready to receive requests from browsers.
- It has a configuration file that tells it what directories are servable; any static Web pages there can be accessed on the Web.
- The configuration file also has the names of specific programs (scripts) that can respond to urls and produce Web pages.
- Any other scripts are useless.

Specific scripts to respond to urls

```
<VirtualHost 169.237.5.130:10002>
ServerName pc110.cs.ucdavis.edu
DocumentRoot /var/www/amenta
WSGIScriptAlias /django /var/www/amenta/mysite/mysite/apache/django.wsgi
WSGIScriptAlias /hw2 /var/www/amenta/hw2/hw2.wsgi
Alias /static/ /var/www/amenta/mysite/mysite/static/
<Directory /var/www/amenta/mysite/mysite/static/>
        Order deny,allow
        Allow from all
</Directory>
</VirtualHost>
```

- Means "any url in directory /hw2 should be supplied by script hw2.wsgi"

Hello World wsgi

- A wsgi script is a Python module in a very specific format.

```
def application(environ, start_response):
    start_response('200 OK',
                   [('Content-Type', 'text/plain')])
    return ['Hello World\n']
```

- function must be named application
- environ is a dictionary
- start_response is a function
- The return value is a…?

The environ dictionary

- We can print it out…

```
    output = str(environ)
    output = "The environment: \n"+output
    return [output]
```

'PATH_INFO': '/tuba.html' - lets you know what the url was that got us into this mess

'QUERY_STRING': 'time=noon&lunch=late'

Grab these!

urlCalled = environ["PATH_INFO"]

query = environ["QUERY_STRING"]

- We can use this to simulate a whole Web app
- Each url can be parsed and handled differently

The start_response function

```
def application(environ, start_response):
    start_response('200 OK',
                   [('Content-Type', 'text/plain')])
    return ['Hello World\n']
```

- Sets up the http header for the Web page in the response.
- Begin with an http status code (200, if we will succeed in returning a Web page, and 404 if not)
- Then other parts of header.

Output some html

```
def application(environ, start_response):
    start_response('200 OK',
                   [('Content-Type', 'text/html')])

    output = """
    <!doctype HTML>
    <html>
    <head> <title>Hello world</title> </head>
    <body> <h1>Hello world!</h1> </body>
    </html> """

    return [output]
```

Working on the server…

- …is not so easy since you're doing it remotely, in Linux.
- I'm working using the gnu emacs text editor, which is sometimes taken to stand for: Generally Not Used, Except by Middle-Aged Computer Scientists.
- Maybe easier to work on your own machine, and test every now and then on the server.

Working local

- Really simple approach
- We'll give you a main program that lets you put in urls and, for each one, calls your application function, and writes out a file containing a Web page.
- You can then look at the Web pages with the browser or the text editor and see if they are what you want.
- Test on the server only when it looks good.