Lecture 5 – HTTP & Django

Web Application Development

January 27, 2015

Jeffrey L. Eppinger
Professor of the Practice
School of Computer Science

Lecture Schedule – 1st Half

(subject to change)

#1	Intro	#9	Django Templates
#2	HTML & CSS	#10	Images
#3	JavaScript & DOM	#11	AJAX
#4	CSS Frameworks	#12	jQuery
#5	HTTP & Django	#13	Databases
#6	Cookies & Sessions	#14	Cloud Deployment
#7	Django Models	#15	SSL
#8	Transactions	#16	Project Proposals

Agenda

→ Course Administration

HTTP

Django

- Python

Homework

Super Bowl Office Hours

Sunday office hours have been moved to the following times:

- Shannon: 1pm to 3pm
- Divya: 3pm to 5pm
- (Same location: WEH 5120)

HW#1 Grades Are Out

…but can you find them?

- We have created a “grades” branch in your student repo
- Easiest way to see grades in a web browser
  - Visit GitHub.com
  - Go to your student repo (for this course)
  - Click on the down arrow besides “branch: master”
  - Select the “grades” branch
- Please do not put changes into this branch
  - The grades on GitHub are a copy from our master directory
- If questions, first contact TA that graded your HW
  - The name of this TA as the bottom of the file

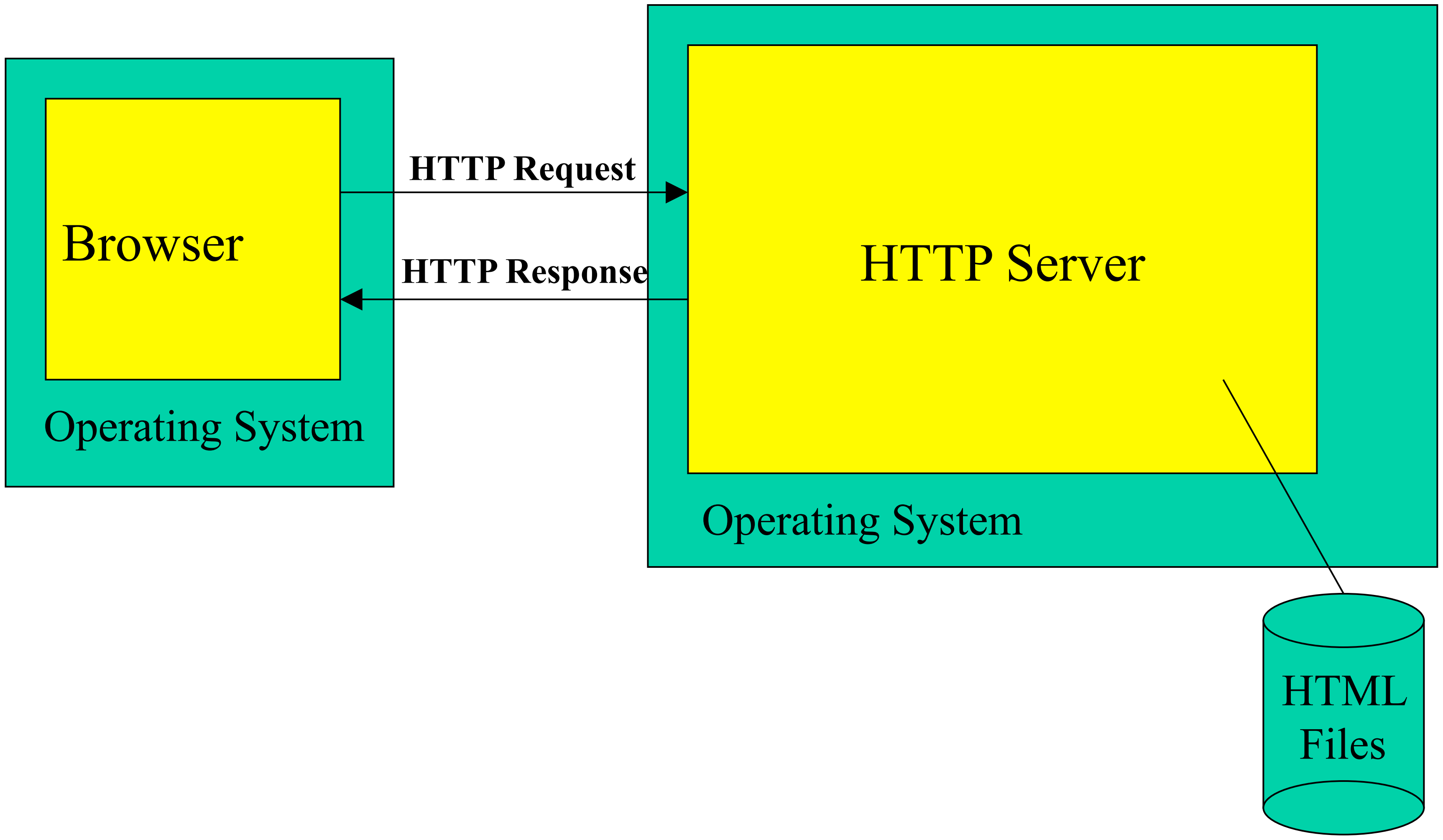
Updated Late Policy

- ✓ HW due on Mondays at 11:59pm
- ✓ No penalty if turned in by Wednesday at 11:59pm
  - But late days are tracked and will delay your signup for project demo
- Penalty if turned in by Thursday at 11:59pm
  - And late days are tracked and will delay your demo signup
  - No need to notify us in advance
- If you want to turn it in after Thursday, you must see the professor after any lecture
  - Late days & penalties apply
  - Please don’t request additional time via e-mail
  - Additional time will not be granted if your HW has already been graded

Agenda

- ✓ Course Administration
- → HTTP
- Django
  - Python
- Homework

Serving Static Content



The HTTP Protocol

Network format for requesting/receiving data from Web

- URI specifies what resource is being accessed
- Connection is TCP protocol on port 80 (by default)
- Request method specified with text command
  - Safe Methods have no side-effects (or aren't supposed to…)
    - GET, HEAD, TRACE, and OPTIONS
  - Idempotent Methods have side-effects:
    - PUT and DELETE (and aren't exactly idempotent)
  - Update Method:
    - POST (although GET is commonly used)
  - Parameters can be passed in GET & POST method
- Header lines follow request line (in text)

HTTP Get Request Format

```
GET <identifier>?<query-string> HTTP/<version>
<header-name>: <header-value>
<header-name>: <header-value>
...
<header-name>: <header-value>
<blank-line>
```

HTTP Get Request Example

```
GET /index.html HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg,
  image/pjpeg, application/x-shockwave-flash,
  application/vnd.ms-excel, application/vnd.ms-
  powerpoint, application/msword, */*
Accept-Language: es-us,en-us;q=0.5
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0;
  Windows NT 5.1; SV1; .NET CLR 1.1.4322)
Host: localhost
Connection: Keep-Alive
```

HTTP Get Request Example w/Param

```
GET /hello.html?name=Barack HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg,
  image/pjpeg, application/x-shockwave-flash,
  application/vnd.ms-excel, application/vnd.ms-
  powerpoint, application/msword, */*
Accept-Language: es-us,en-us;q=0.5
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0;
  Windows NT 5.1; SV1; .NET CLR 1.1.4322)
Host: localhost
Connection: Keep-Alive
```

HTTP Post Request Format

```
POST <identifier> HTTP/<version>
<header-name>: <header-value>
...
<header-name>: <header-value>
Content-Length: <message-length>
<header-name>: <header-value>
...
<header-name>: <header-value>
<blank-line>
<message-body>
```

Network Addressing

- Contact a computer using a network address
  - Low-level, hardware address: MAC Address
  - Routable address: IP Address
  - High-level, logical address: DNS Hostname
- Contact an application (on a computer) using a port
  - There are standard ports on which apps listen
    - E.g., Telnet (23), SMTP (25), RDP (3389)
    - For deployment: HTTP (80), SSL (443), MySQL (3306)
    - For development: HTTP (8000), SSL (8443)

You Can Be the Browser

- Telnet to a webserver
- Enter HTTP requests
- Example using HTTP 1.0:

```
> telnet www.cmu.edu 80
GET /index.shtml
```

- Example using HTTP 1.1:

```
> telnet www.cmu.edu 80
GET /index.shtml HTTP/1.1
Accept: */*
Host: www.cmu.edu
Connection: Keep-Alive
```

<blank-line>

HTTP Server - Runtime

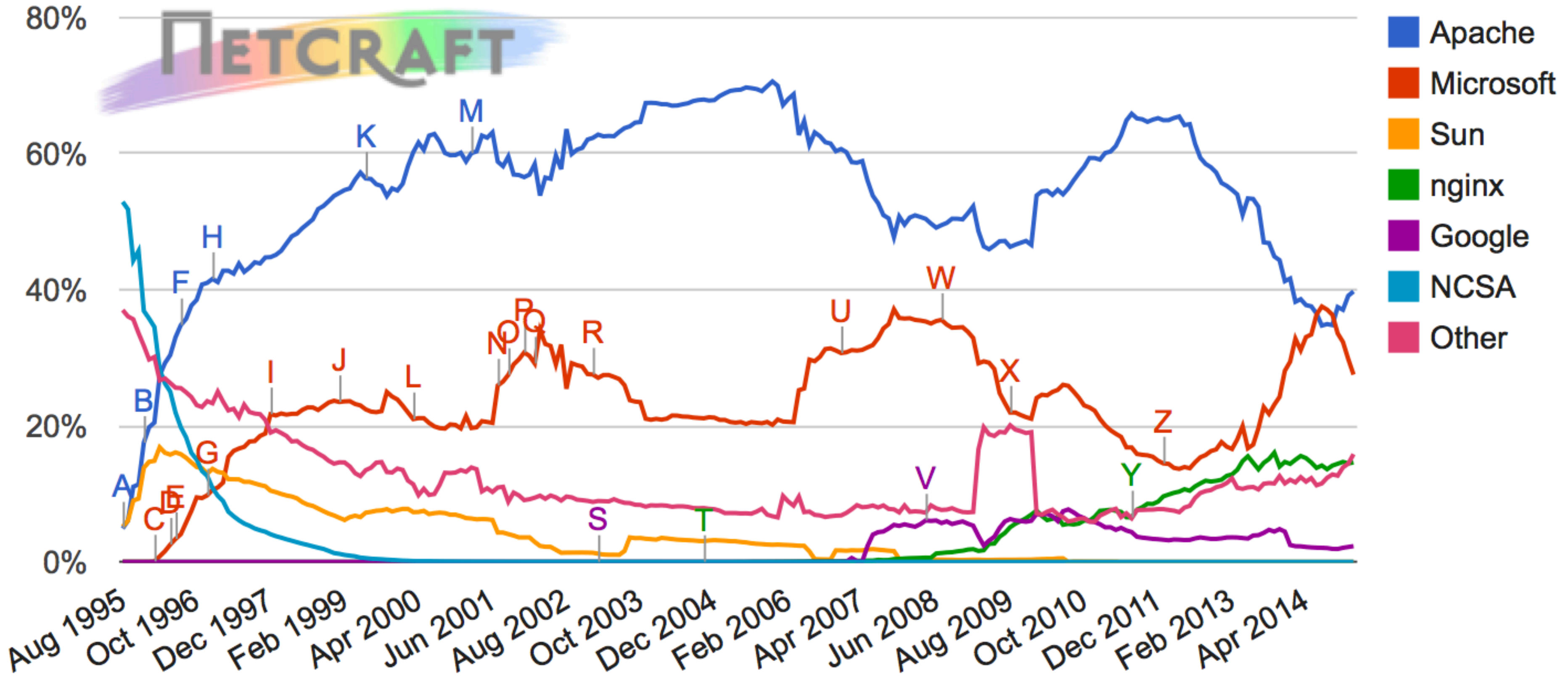
- Much complexity due to size of HTTP specification
- Basic run-time structure (the classic server structure)

```
while (true) {
    request = readHttpRequest(…);
    response = processHttpRequest(request);
    sendHttpResponse(…,response);
}
```

- Obvious issues: performance, parallelism, portability, *security*

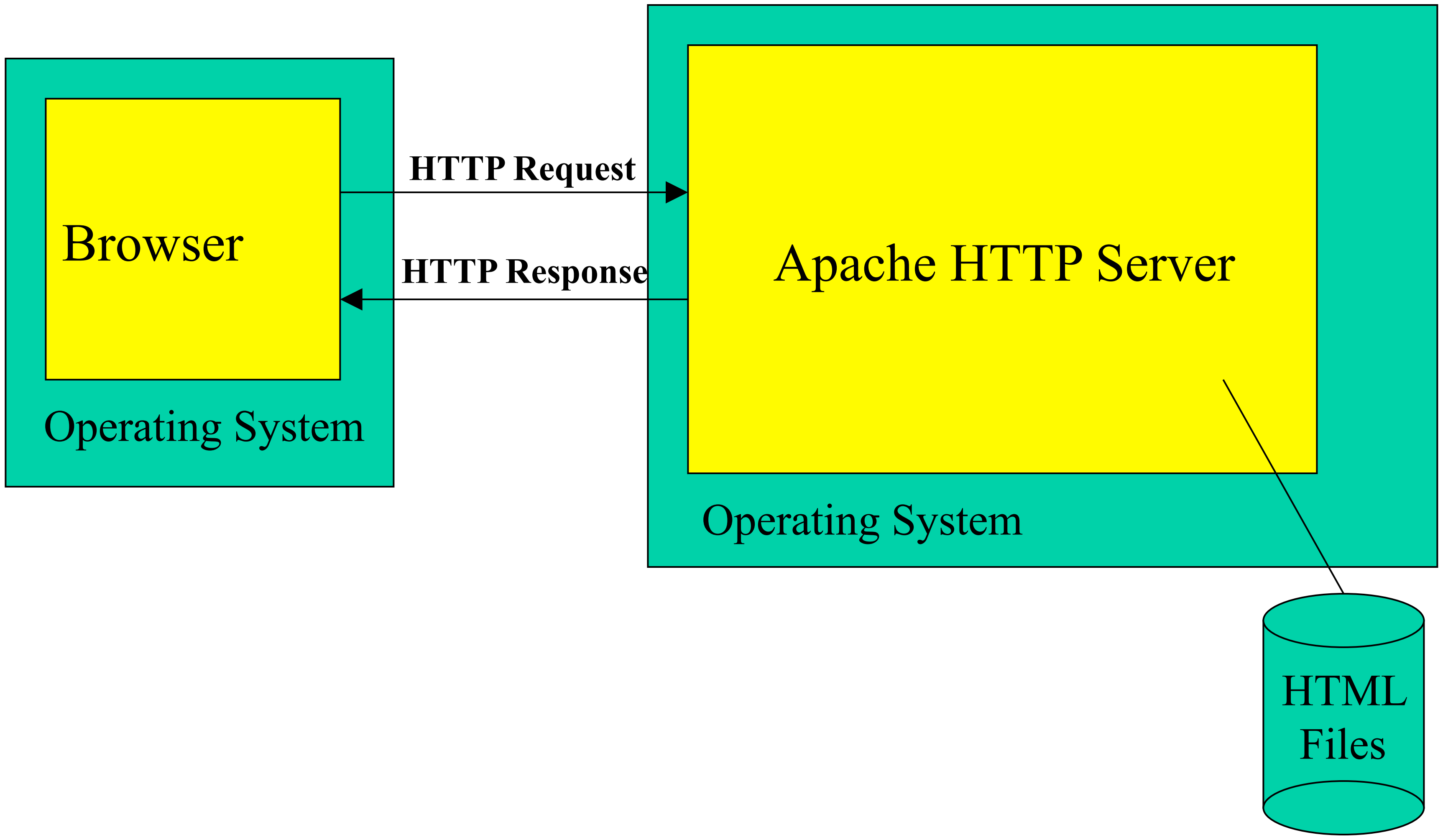
Server Market Share

Study of one billion websites



Source: Netcraft LTD (http://news.netcraft.com/archives/web_server_survey.html)

Apache HTTP Server



Apache HTTP Server

- Most popular web server
- Open Source (download from www.apache.org)
- Key Directory Information
  - By default: `c:\Program Files\Apache Group\Apache\`
    - Document root: `htdocs\`
    - Audit root: `logs\`
    - Configuration root: `conf\`
    - CGI-bin root: `cgi-bin\`
  - Organization of documents and scripts important for long-term management

HTTP Server – Administration

- Server name, email addresses
- Locations of documents
- IP addresses and ports
- Timeouts, maximum length requests
- Processing options (e.g., CGI enabled?)
- Cache handling
- Automated directory display
- Authentication/Authorization
- Audit/Logging
- Error reporting
- Systems management

You Can Be the HTTP Server

- How long to whip up a simple Java Web Server?
  - About one hour + another hour to debug it
- How much code would that take?
  - Less than 200 Lines
  - Only handles GET requests
  - Doesn’t send back correct error messages
- See: SimpleHttpServer.java

Agenda

✓ Course Administration

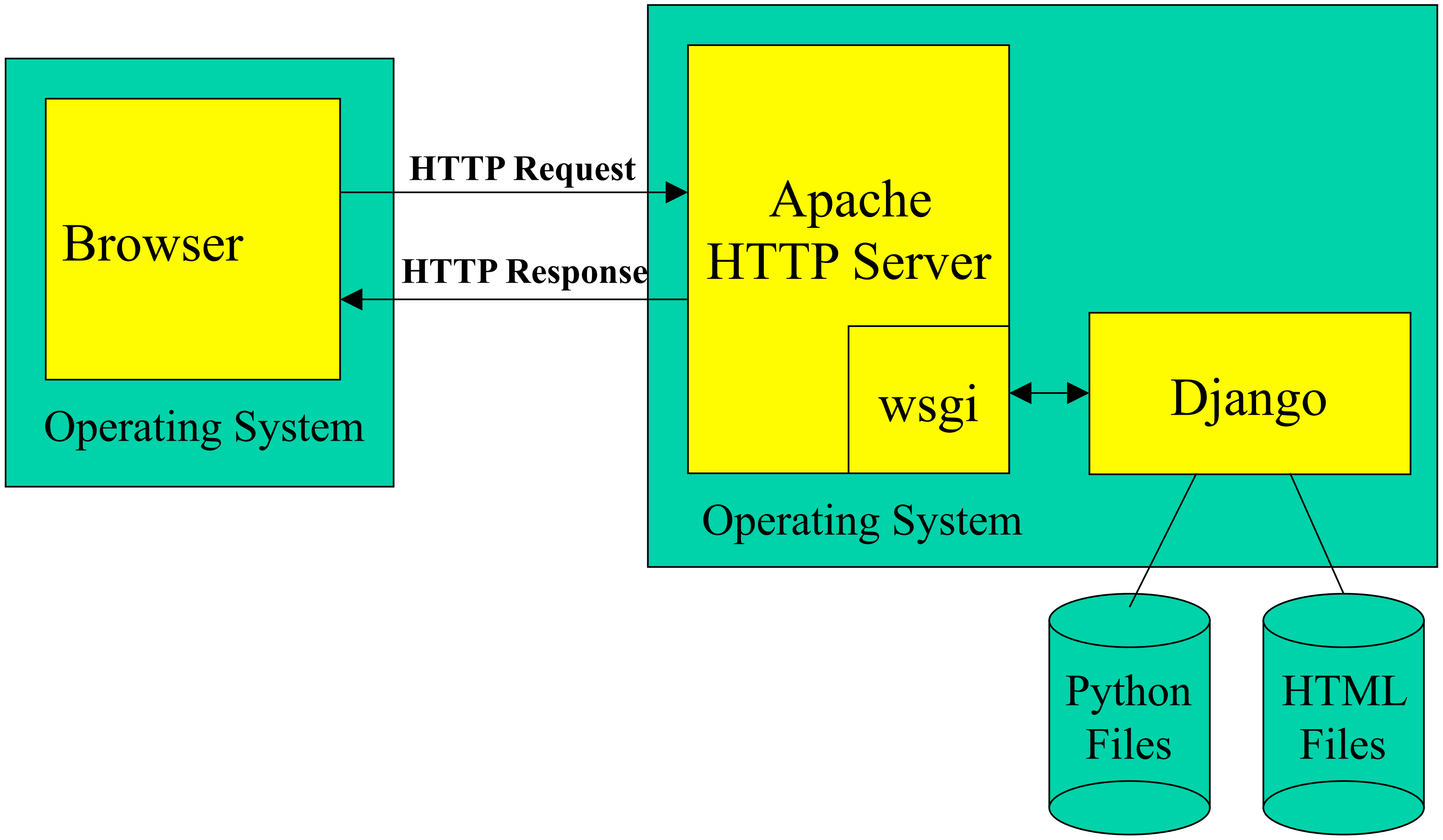
✓ HTTP

→ Django

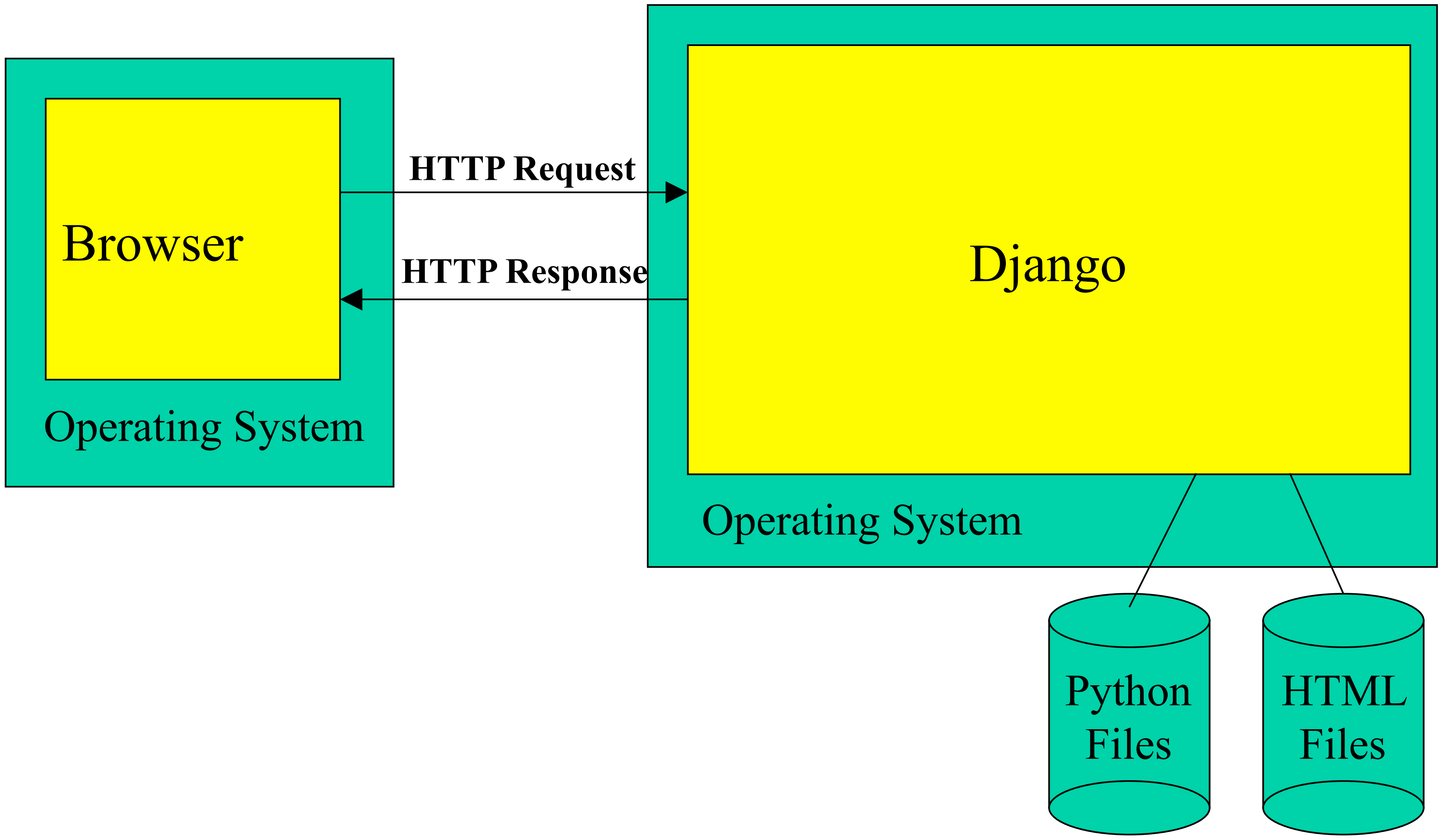
  - Python

  Homework

Typical Django Deployment



Django Development Server



Hello World

```
from django.http import HttpResponse

def hello_world(request):
    html="""
        <!DOCTYPE HTML>
        <html>
          <head>
              <meta charset="utf-8">
              <title>Hello World</title>
          </head>
          <body>
              <h1>Hello, World!</h1>
          </body>
        </html>
    """
    return HttpResponse(html)
```

Getting to Code

- Django Project
  - Project directory (webapps)
    settings.py – initialization
    urls.py – provides the connection to the actions in views.py
    wsgi.py – used when deploying via Apache HTTP Server
  - App directory (intro)
    views.py – code to implement the actions
    models.py – database models

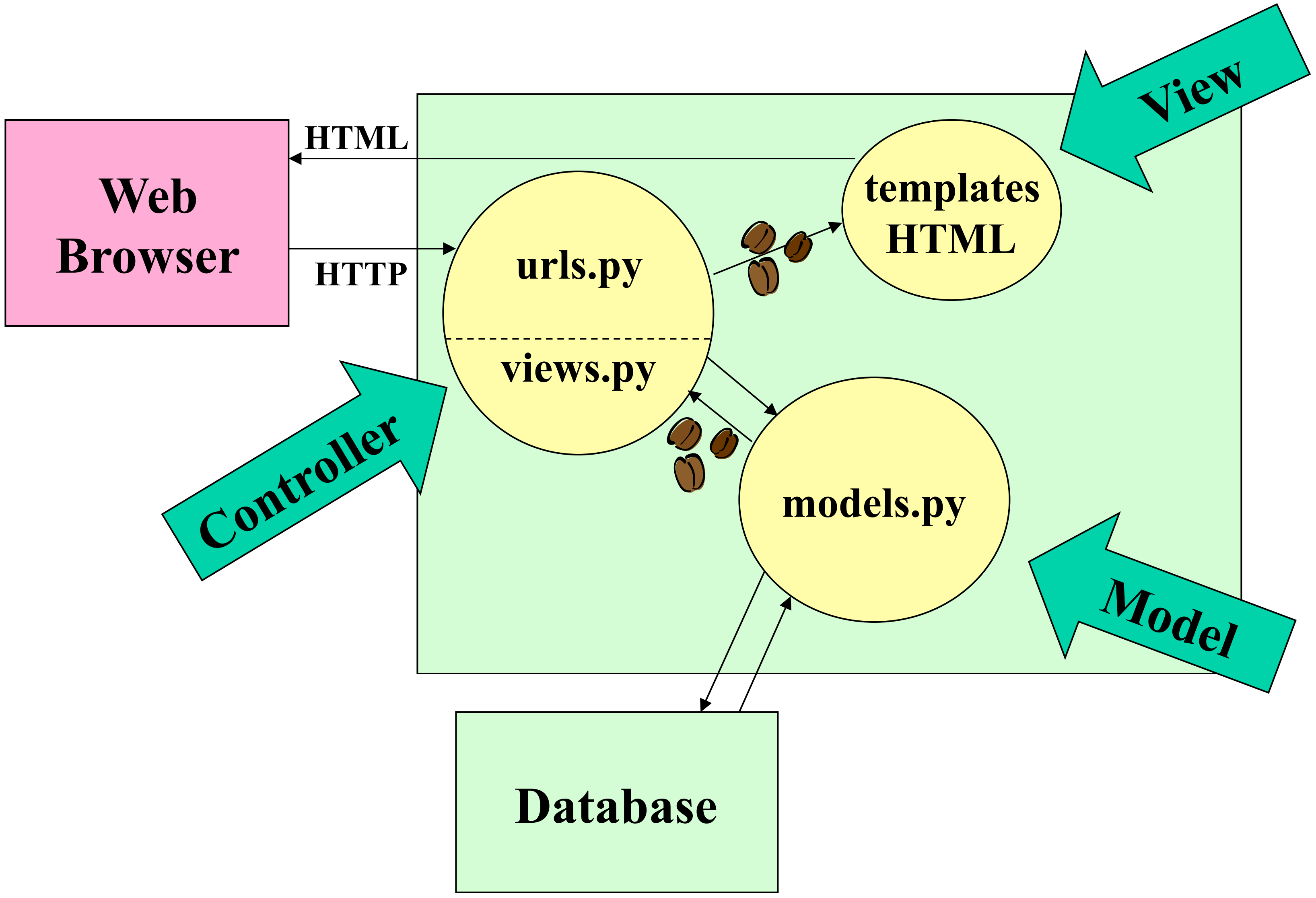
Django Templates

- Located in the app’s template directory
  (intro/templates)
- HMTL Files…
- …augmented by Django Template Language
  - We will cover this in more detail in a future lecture
  - But check out “greet.html” in today’s example

More Examples, with Templates

- Hello World with Template
- Greet
- Home Page

Model-View-Controller Architecture



Separation of Concerns

MVC gives us “Separation of Concerns”

Different people can work on each part

- DB people can build the models (models.py)
- App developers can build controller actions (in views.py)
- Web designers can build the Views (in templates/…/*.html)

Agenda

- ✓ Course Administration
- ✓ HTTP
- → Django
  - → Python
- Homework

Know Python

- We’ll be using Django is a Web App Framework
  - Runs on Python Programming Language
  - We’ll be using Python 2.7 for this course
- Know how to write small Python programs easily
  - Know all the basic language constructs
    - Especially loops, lists, and dictionaries (maps)

Python Resources

- Python Official Home Page: http://www.python.org
  - Install Python 2.7.x (currently 2.7.9)
  - If you have a MAC is should already by installed
    - Check by simply executing the "python" command in a terminal
- Google Python class
  - https://developers.google.com/edu/python/
    - Read Introduction through Files
    - Complete the Basic Exercises
- Help pages for some Python built-in features:
  - http://docs.python.org/2/library/stdtypes.html
  - http://docs.python.org/2/library/functions.html

Django Website

https://djangoproject.com

- Installation instructions
- Documentation
- Tutorial

Homework #3

- Should be posted tonight (tomorrow at the latest)
- Make your calculator from HW#1 function using Django
- Install Python 2.7, if you don't have it
  - Should just be there on MACs
- Install Django 1.7
- Get today's example running
  - Clone the "django-intro" repo
    - Do it outside your student repo
  - python manage.py runserver
  - Visit http://localhost:8000
- Create your HW#2 project in your student repo
  - Copy your HTML files into the template

Next Class

- Hidden Fields, Cookies and Sessions
  - You'll to know this to maintain your calculator's context
- Bring Laptops
  - We may have "in-class" exercises