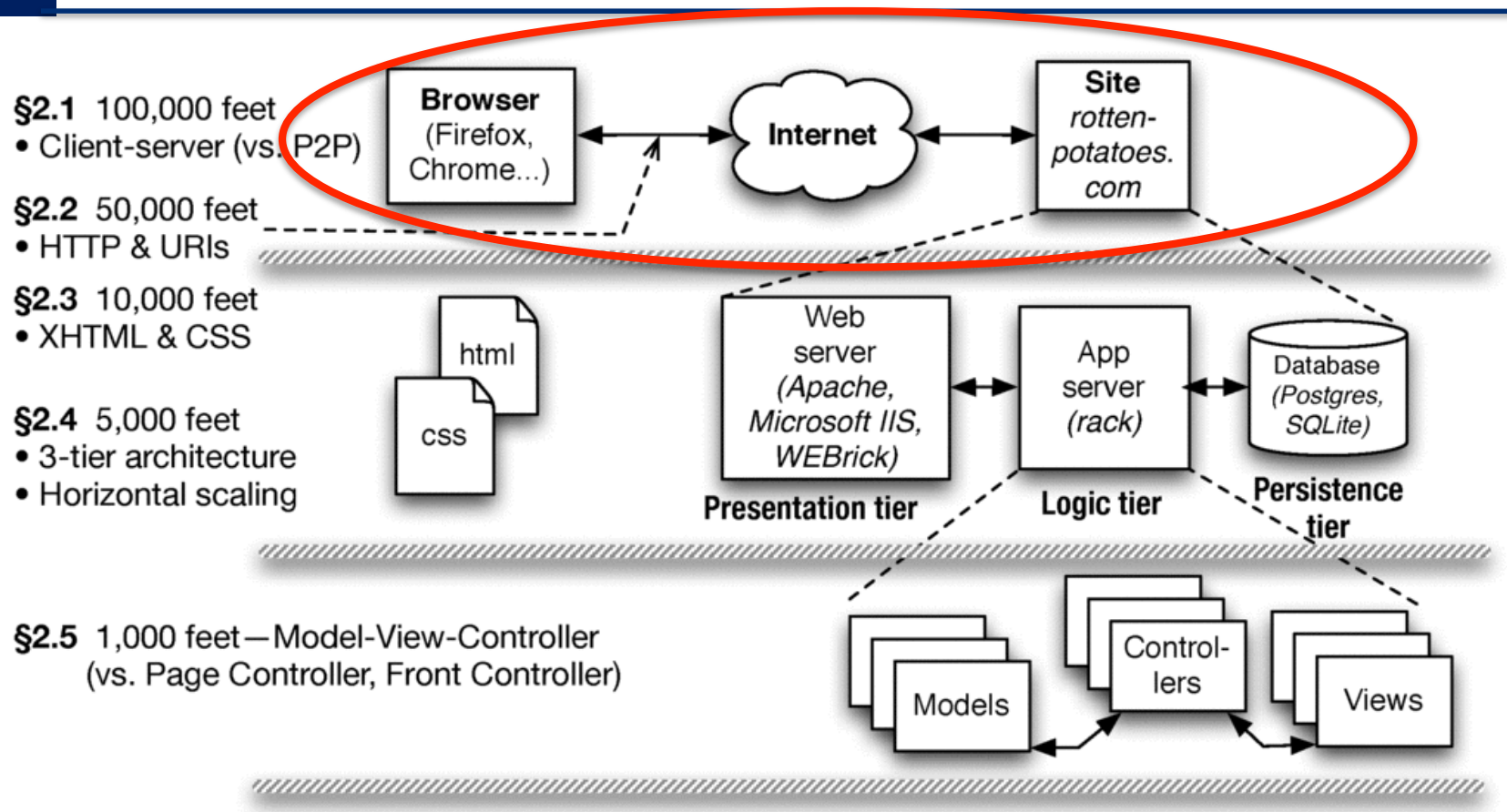
The Web as a Client-Server System; TCP/IP intro

Engineering Software as a Service
§2.1–2.2

Armando Fox



§2.6 500 feet: Active Record models (vs. Data Mapper)
§2.7 500 feet: RESTful controllers (Representational State Transfer for self-contained actions)
§2.8 500 feet: Template View (vs. Transform View)

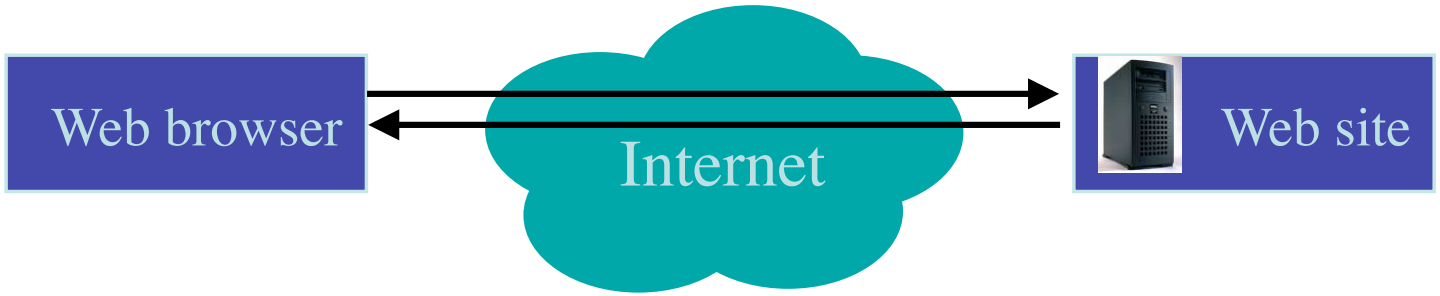
• **Active Record**
• Data Mapper

• **REST**

• **Template View**
• Transform View

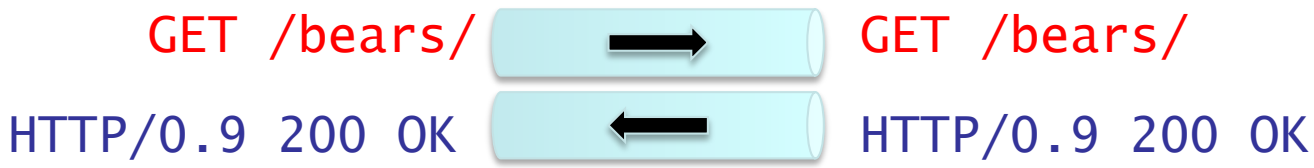
Web at 100,000 feet

- The web is a c*lient/server* architecture
- It is fundamentally *request/reply oriented*



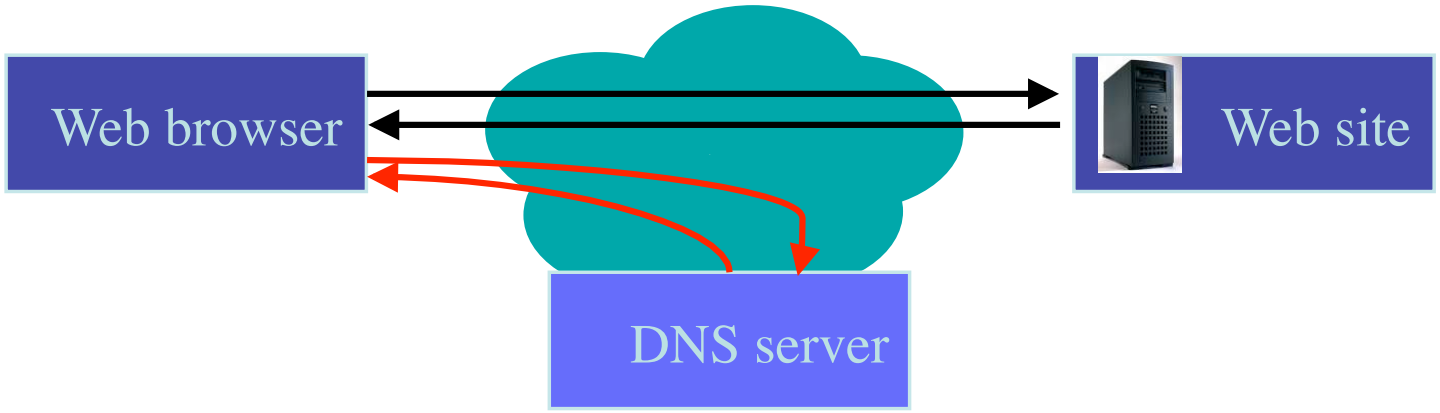
Nuts and bolts: TCP/IP protocols

- IP (Internet Protocol) *address* identifies a physical network interface with four *octets,* e.g. `128.32.244.172`
  - Special address `127.0.0.1` is "this computer", named `localhost`, even if not connected to the Internet!
- TCP/IP (Transmission Control Protocol/Internet Protocol)
  - IP: no-guarantee, best-effort service that delivers *packets* from one IP address to another
  - TCP: make IP reliable by detecting "dropped" packets, data arriving out of order, transmission errors, slow networks, etc., and respond appropriately
  - TCP *ports* allow multiple TCP apps on same computer
- Vint Cerf & Bob Kahn: 2004 Turing Award for Internet architecture & protocols, incl. TCP/IP



Web at 100,000 feet

- The web is a *client/server* architecture
- It is fundamentally *request/reply oriented*
- Domain Name System (DNS) is another kind of server that maps *names* to *IP addresses*



Now that we're talking, what do we say?
Hypertext Transfer Protocol

- an *ASCII-based request/reply protocol* for transferring information on the Web
- *HTTP request* includes:
  - *request method* (`GET`, `POST`, etc.)
  - Uniform Resource Identifier (URI)
  - HTTP protocol version understood by the client
  - *headers*—extra info regarding transfer request
- *HTTP response* from server
  - Protocol version & Status code =>
  - Response headers
  - Response body

HTTP status codes:
2xx — *all is well*
3xx — *resource moved*
4xx — *access problem*
5xx — *server error*

Cookies

- Observation: *HTTP is stateless*
- Early Web 1.0 problem: how to guide a user "through" a flow of pages?
  - use IP address to identify returning user?
    - ✖ public computers, users sharing single IP
  - embed per-user junk into URI query string?
    - ✖ breaks caching
- Quickly superseded by *cookies*
  - Watch: *screencast.saasbook.info*
- Rails manages tamper-evident cookies for you

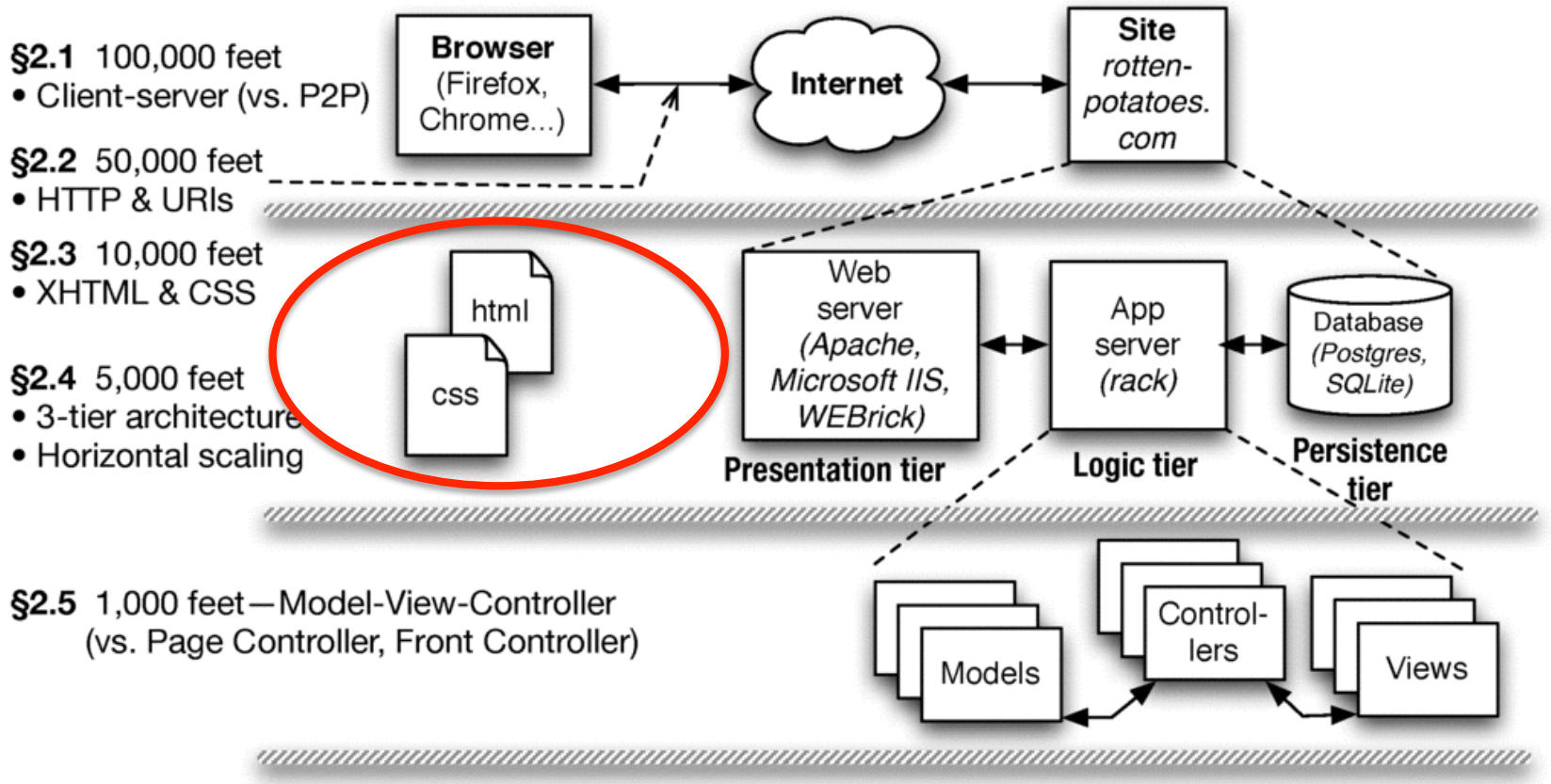
A ____ can create and modify cookies; the ____ is responsible for including the correct cookie with each request

- ☐ Browser; SaaS app
- ☐ SaaS app; browser
- ☐ HTTP request; browser
- ☐ SaaS app; HTTP response

HTML+CSS

Engineering Software as a Service §2.3

Armando Fox



§2.6 500 feet: Active Record models (vs. Data Mapper)
§2.7 500 feet: RESTful controllers (Representational State Transfer for self-contained actions)
§2.8 500 feet: Template View (vs. Transform View)

• **Active Record** • Data Mapper
• **REST**
• **Template View** • Transform View

Introduction

This article is a review of the book Dietary Preferences of Penguins, by Alice Jones and Bill Smith. Jones and Smith's controversial work makes three hard-to-swallow claims about penguins:

First, that penguins actually prefer tropical foods such as bananas and pineapple to their traditional diet of fish

Second, that tropical foods give penguins an odor that makes them unattractive to their traditional predators

```
<h1>Introduction</h1>
<p>
  This article is a review of the book
  <i>Dietary Preferences of Penguins</i>,
  by Alice Jones and Bill Smith. Jones and Smith's
  controversial work makes three hard-to-swallow claims
  about penguins:
</p>
<ul>
  <li>
   First, that penguins actually prefer tropical foods
   such as bananas and pineapple to their traditional diet
   of fish
  </li>
  <li>
   Second, that tropical foods give penguins an odor that
   makes them unattractive to their traditional predators
  </li>
</ul>
...
```

Introduction

This article is a review of the book *Dietary Preferences of Penguins*, by Alice Jones and Bill Smith. Jones and Smith's controversial work makes two hard-to-swallow claims about penguins:

- First, that penguins actually prefer tropical foods such as bananas and pineapple to their traditional diet of fish
- Second, that tropical foods give penguins an odor that makes them unattractive to their traditional predators

...

```
<h1>Introduction</h1>
<p>
This article is a review of the book
<i>Dietary Preferences of Penguins</i>,
by Alice Jones and Bill Smith. Jones
and Smith's controversial work makes
three hard-to-swallow claims about
penguins:
<ul>
<li>
First, ...
```

Hypertext Markup Language

- Document = Hierarchy of *elements*
  - inline (headings, tables, lists, paragraphs)
  - embedded (images, JavaScript)
  - forms—allow user to submit simple input (text, radio/check buttons, dropdown menus...)
- Elements delimited by `<tag>....</tag>`
  - Some have *content:* `<p>Hello world</p>`
  - Some have *attributes:* `<img src="http://...">`
  - `id` and `class` attributes useful for *styling*

Cascading Style Sheets (CSS) separate content from presentation

- `<link rel="stylesheet" href="http://..."/>` (inside `<head>` element): what stylesheet(s) go with this HTML page
- HTML `id` & `class` attributes important in CSS
  - *id* must be ***unique within this page***
  - same *class* can be attached to many elements

```
<div id="right" class="content">
  <p>
   I'm Armando.  I teach CS169 and do
   research in the AMP Lab and Par Lab.
  </p>
</div>
```

CSS Selectors identify specific elements for styling

```
<div class="pageFrame" id="pageHead">
  <h1>
    Welcome,
    <span id="custName">Armando</span>
  </h1>
  <img src="welcome.jpg" id="welcome"/>
</div>
```

- tag name: `h1`
- class name: `.pageFrame`
- element ID: `#pageHead`

both of these match the outer *div* above. Don't do this!

- tag name & class: `div.pageFrame`
- tag name & id: `img#welcome` (usually redundant)
- descendant relationship: `div .custName`
- Attributes *inherit* browser defaults unless overridden

Goal: HTML markup contains no visual styling information

Which CSS selector will select ***only*** the word “bar” for styling:

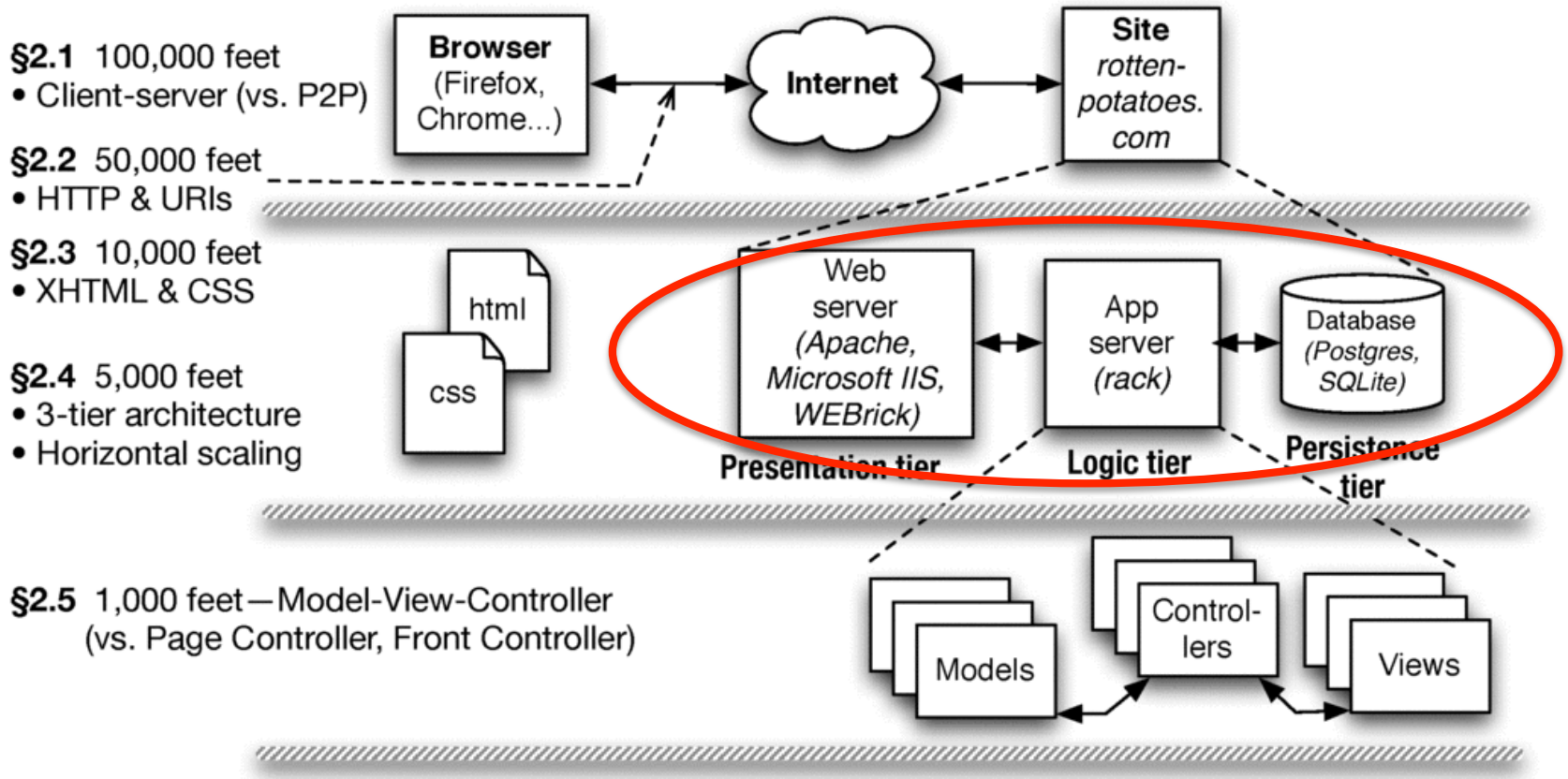
```
<p class="myClass">foo,
 <span class="myClass">bar<span></p>
```

- ☐ `span.myClass`
- ☐ `p  .myClass`
- ☐ `.myClass  span`
- ☐ All of these

3-tier shared-nothing architecture & scaling

Engineering Software as a Service §2.4

Armando Fox



§2.6 500 feet: Active Record models (vs. Data Mapper)
§2.7 500 feet: RESTful controllers (Representational State Transfer for self-contained actions)
§2.8 500 feet: Template View (vs. Transform View)

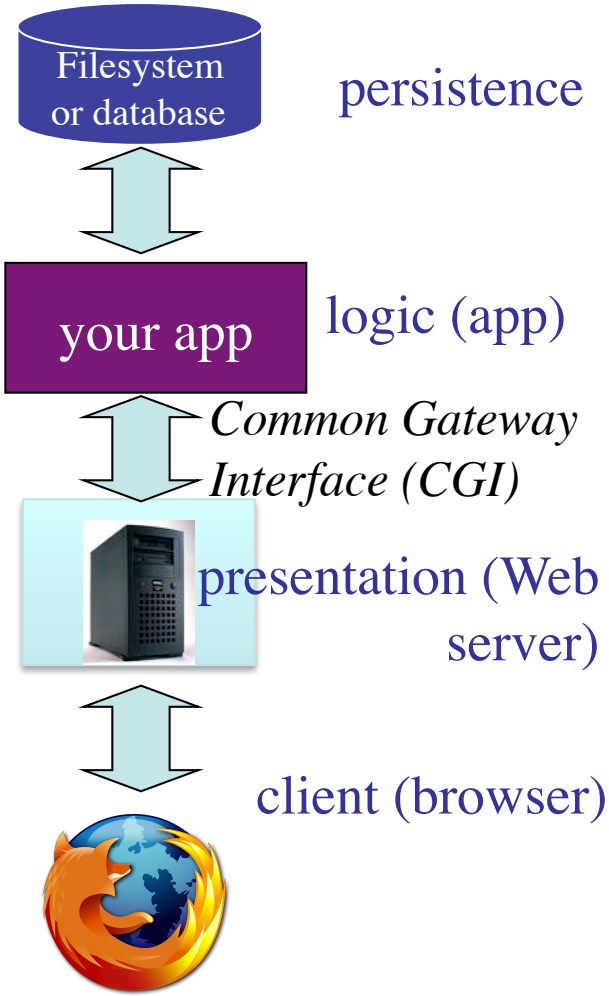
- **Active Record**
- Data Mapper
- **REST**
- **Template View**
- Transform View

Dynamic content generation

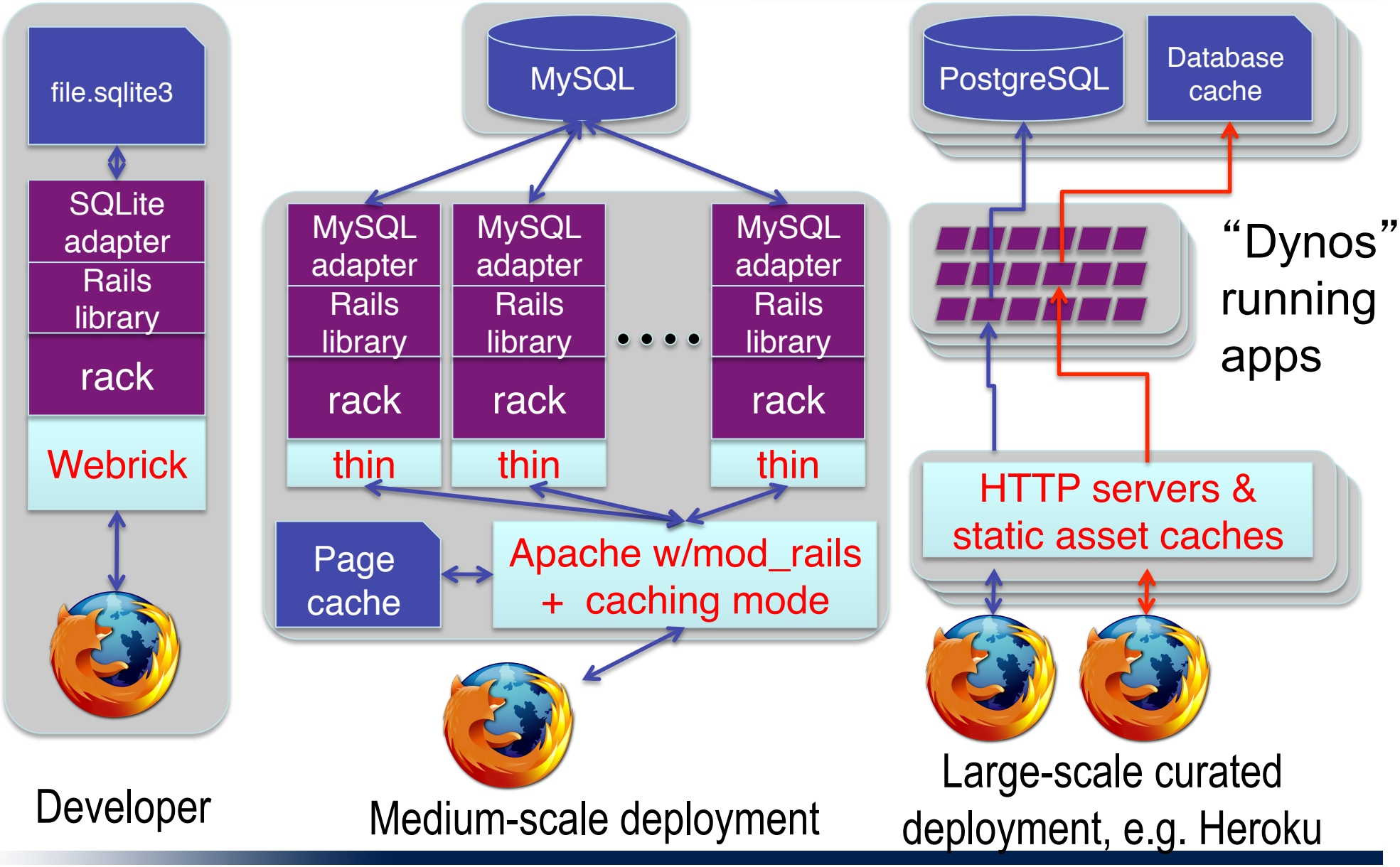
- In the Elder Days, most web pages were (collections of) plain old files
- But most interesting Web 1.0/e-commerce sites *run a program* to generate each "page"
- Originally: templates with embedded code "snippets"
- Eventually, code became "tail that wagged the dog" and moved out of the Web server

Sites that are really programs (SaaS)

- How do you:
  - "map" URI to correct program & function?
  - pass arguments?
  - invoke program on server?
  - handle persistent storage?
  - handle cookies?
  - handle errors?
  - package output back to user?
- *Frameworks* support these common tasks

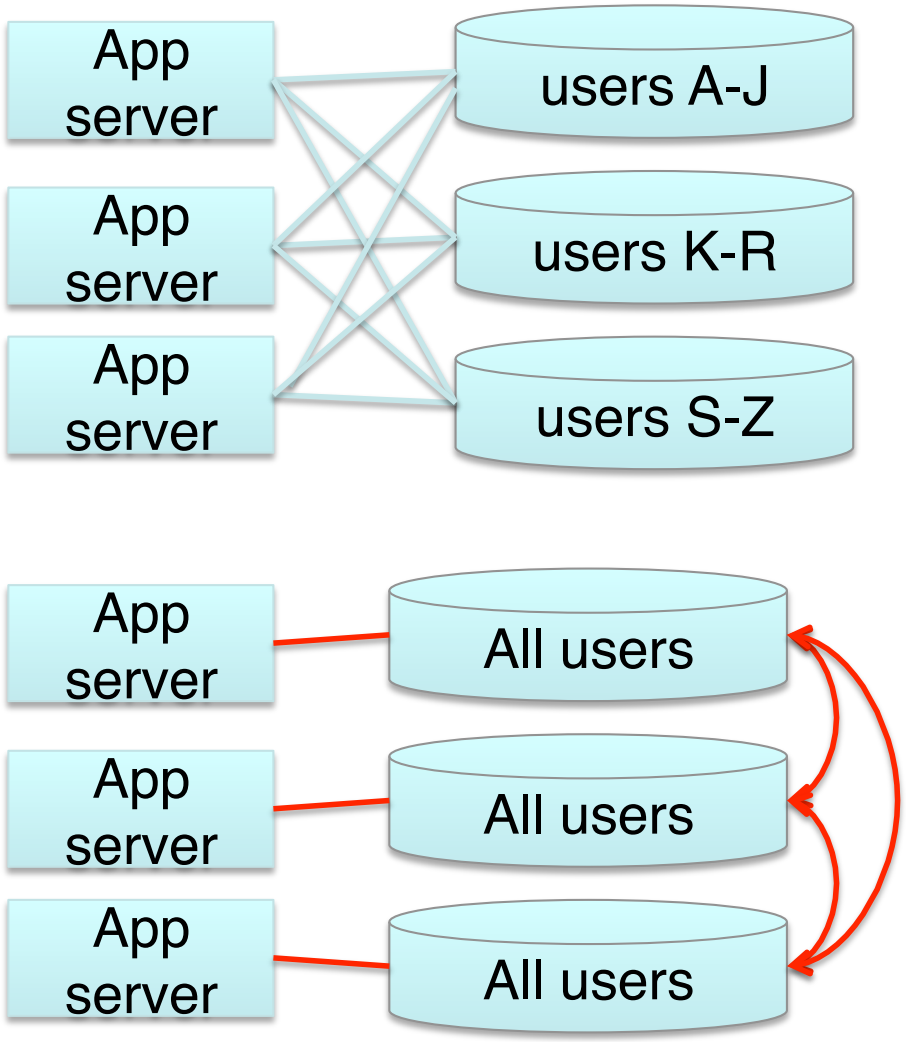


Developer environment vs. medium-scale deployment



Sharding vs. Replication

- Partition data across independent "shards"?
  - + Scales great
  - – Bad when operations touch >1 table
  - – Example use: user profile
- Replicate all data everywhere?
  - + Multi-table queries fast
  - – Hard to scale: writes must propagate to all copies => temporary *inconsistency* in data values
  - – Example: Facebook wall posts/"likes"



Summary: Web 1.0 SaaS

- Browser *requests* web resource (URI) using HTTP
  - HTTP is a simple request-reply protocol that relies on TCP/IP
  - In SaaS, most URI's cause a program to be run, rather than a static file to be fetched
- *HTML* is used to encode content, *CSS* to style it visually
- *Cookies* allow server to track client
  - Browser automatically passes cookie to server on each request
  - Server may change cookie on each response
  - Typical usage: cookie includes a *handle* to server-side information
  - That's why some sites don't work if cookies are completely disabled
- *Frameworks* make all these abstractions convenient for programmers to use, without sweating the details
- ...and help map SaaS to 3-tier, shared-nothing architecture

Match the terms:
(a) presentation tier, (b) logic tier,
(c) persistence tier

- ☐ (a) Apache web server (b) Rack+Rails (c) Relational database
- ☐ (a) Firefox (b) Apache web server (c) PostgreSQL
- ☐ (a) Microsoft Internet Information Server (b) Rack+Rails (c) Apache web server
- ☐ (a) Firefox (b) Microsoft Internet Information Server (c) MySQL