Web development with Ruby on Rails

Unit 2: HTTP, HTML and CSS

Homework on Blackboard

Discussion on Piazza

http://piazza.com/upenn/fall2012/cis196

My Office hours, Wednesday morning

6th Floor Levine Lounge

HTTP

- How computers on the internet talk to eachother
- How your browser talks to the internet
- How your browser talks to your Rails app

A GET

- Browser: Ohai, can I get an expense list?
- Server: Sure, one sec
- Server: Okay, here it is, it's an HTML file, btw.
- Server: (attachment: expenses.html)

```
> GET /expenses HTTP/1.1

HTTP/1.1 200 OK
Content-Type: text/html;charset=utf-8
Content-Length: 2959
Connection: keep-alive
Server: thin 1.3.1

<!DOCTYPE html>
...
```

A POST

- Browser: I'm back, can you keep this somehere safe for me?
- Browser: It's a new expenses document.
- Browser: (attachment: expense form data)
- Server: Sure, got it.
- Server: Do you want to see a list of all the expense documents?

```
> POST /expenses HTTP/1.1
> Content-Type:
>   application/x-www-form-urlencoded
> Content-Length: 39
>
> description=Lunch+with+mom&amount=2.

HTTP/1.1 302 Found
Location: /expenses
```

Also DELETE and PUT

Headers

- Tell the server about the request
- Tell the browser about the response

```
> POST /expenses HTTP/1.1
> X-This-Is: A request header
>
> (content)

HTTP/1.1 200 Ok
X-This-Is: A response header

(content)
```

Cookies

- A secret that the browser and server share
- Server creates them
- Sends as a Set-Cookie header
- Browser only send them back to the server they came from
- Sends as a Cookie header

Cookies

- Useful mainly for identification
- Also can store small amounts of user data

- Server: Here's your expense list
- Server: oh, let's make "cupcake" our secret word
- Browser: Thanks! fun, it's like we're spys :)

```
> GET /expenses HTTP/1.1

HTTP/1.1 200 OK
Content-Type: text/html;charset=utf-8
Content-Length: 2959
Connection: keep-alive
Server: thin 1.3.1
Set-Cookie: secret=cupcake

<!DOCTYPE html>
...
```

- Browser: Hey it's me again, can I get the latest expense list?
- Server: prove it
- Browser: cupcake, now give me the list
- Server: sorry, having a paranoid day. Here you go

```
> GET /expenses HTTP/1.1

HTTP/1.1 403 Forbidden

> GET /expenses HTTP/1.1
> Cookie: secret=cupcake

HTTP/1.1 200 OK
Content-Type: text/html;charset=utf-8
Content-Length: 2959
Connection: keep-alive
Server: thin 1.3.1

<!DOCTYPE html>
...
```

RESTful web design

- Using HTTP to it's full capabilities
- GET, POST, PUT, DELETE
- Using headers to talk about the data
- Treating URLs as "resources"

Rails REST

```
GET    /things   -> a list of things
GET    /things/1 -> a thing
POST   /things   -> make a new thing
PUT    /things/1 -> update an old thir
DELETE /things/1 -> delete a thing
```

Let's make a resource

```
cd plusdollar
rails generate scaffold \
               pledge \
               issue_url:string \
               issue_title: string \
               amount:decimal
rake db:migrate
rails server
```

Watching it happen

- Get Chrome
- Open the Network panel
- Click "Record"
- Start clicking!

HTML

The web's Word doc

Boilerplate

```
<!DOCTYPE html>
<html>
  <head>
    <title>My awesome page</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

Things that do stuff

Links

```
<a href="someotherpage.html">
  Click me
</a>
```

Forms

```
<form action="someotherpage.html"
      method="post">
  <input type="text" name="a_field" />
  <input type="submit"
         value="A Button!" />
</form>
```

Make it pretty with CSS

```
<p>Some stuff</p>
```

Some stuff

```
<div class="warning">
  <p>Some stuff</p>
</div>
```

```
.warning { color: red; }
```

Some stuff

Can use id="blah" and #blah

Or mix classes

class="big warning" and .big.warning

More at reference.sitepoint.com

Or use a framework

http://twitter.github.com/bootstrap

http://foundation.zurb.com

http://thoughtbot.com/neat/

http://960.gs/

http://thesquaregrid.com/

Homework

Put a link on your index.html to your blog, github or twitter profile

Put a form on your index.html with at least an input field and a button

(doesn't have to post to anywhere)

Style up the page either by hand or with a framework

Bonus (3pts): Make a scaffolded resource in your rails app, interact with it

http://guides.rubyonrails.org/getting_started.html

#philly.rb IRC channel on freenode.net for help.