Day one

June, Monday 22nd


Fun
stuff
today!

Day one, Monday 22nd

- HTTP & Sinatra routes & redirects
- Sinatra views & ERB templating

IRON HACK

About HTTP

Tell me stuff

- States for *HyperText Transfer Protocol* (HTTP)

- A protocol is a set of rules that determine how information is transferred and treated

- First proposal in 1989 (>25 years already!) by Sir Tim Berners-Lee

Tell me (more) stuff

- Stateless!
- Content-independent
- Based on communication between nodes

Request & response

- The client performs requests

- And the server answers with responses

- That's it

A graphic way

Request ironhack.com

Response with the site content

IRON HACK

Verbs

- They indicate the desired action for the request
- Their purpose is semantic, in order for the server to answer properly
- Predetermined, a total of 9 exist

(a list of) verbs

- GET
- POST
- PUT
- DELETE
- HEAD
- TRACE
- OPTIONS
- CONNECT
- PATCH

Status codes

- They indicate how the request went
- They are composed by both a number and a phrase
- Standard, so communication is better and more efficient

(a table of) status codes

- 1xx: Message received, doing stuff
- 2xx: Everything done
- 3xx: Go to this other place
- 4xx: You fucked up
- 5xx: I fucked up

A raw example (header)

```
 curl -v example.com
* Rebuilt URL to: example.com/
* Hostname was NOT found in DNS cache
*   Trying 93.184.216.34...
* Connected to example.com (93.184.216.34) port 80 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.37.1
> Host: example.com
> Accept: */*
>
< HTTP/1.1 200 OK
< Accept-Ranges: bytes
< Cache-Control: max-age=604800
< Content-Type: text/html
< Date: Tue, 20 Jan 2015 14:52:31 GMT
< Etag: "359670651"
< Expires: Tue, 27 Jan 2015 14:52:31 GMT
< Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
* Server ECS (fll/0761) is not blacklisted
< Server: ECS (fll/0761)
< X-Cache: HIT
< x-ec-custom-error: 1
< Content-Length: 1270
<
```

IRON HACK

A raw example (body)

```
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 50px;
        background-color: #fff;
        border-radius: 1em;
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
        body {
            background-color: #fff;
        }
        div {
            width: auto;
            margin: 0 auto;
            border-radius: 0;
            padding: 1em;
        }
    }
```

IRON HACK

Quick intro

Web frameworks

- A set of tools for a specific language that abstract common actions related to web…

- …so you don't have to implement it yourself

- They ease routing, HTTP responses, templating… you name it

Sinatra routing

This is Sinatra

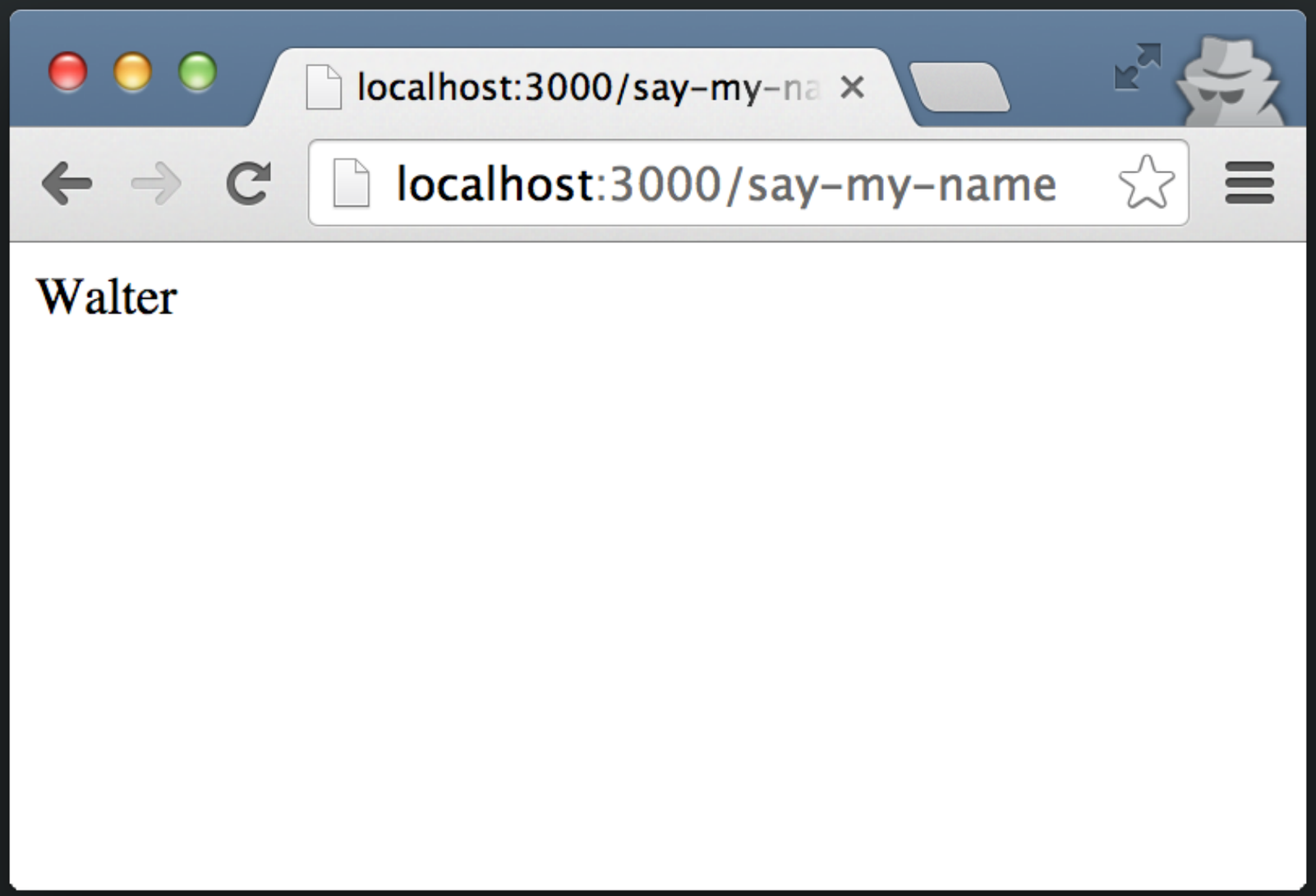
- Named after a New-York-based, Mafia-near, awesome-singer, smiley guy

- It's a minimal web framework: listens through a port, and answers stuff

“Say my name”

```ruby
require 'sinatra'

set :port, 3000
set :bind, '0.0.0.0'

get '/say-my-name' do
  'Walter'
end
```

THE NAME IS SAID!!!1111



Reload me, baby

gem install sinatra-contrib

```ruby
require 'sinatra'
require 'sinatra/reloader'

set :port, 3000
set :bind, '0.0.0.0'

get '/say-my-name' do
  erb :index
end
```

(please reload the app)

IRON HACK

More Sinatras…

sinatrarb.com

github.com/sinatra/sinatra

And a tool: curl

brew install curl

(Linux has it already)

Not these ones :(

```
 curl localhost:3000/say-my-name
Walter%
```

more info: man curl

LIVE CODING FROM BCN!!!!111

Let's say hi from Sinatra

LET'S GET READY TO RUMBLE

Exercises SL1 & SL2

IRON HACK

ERB templating

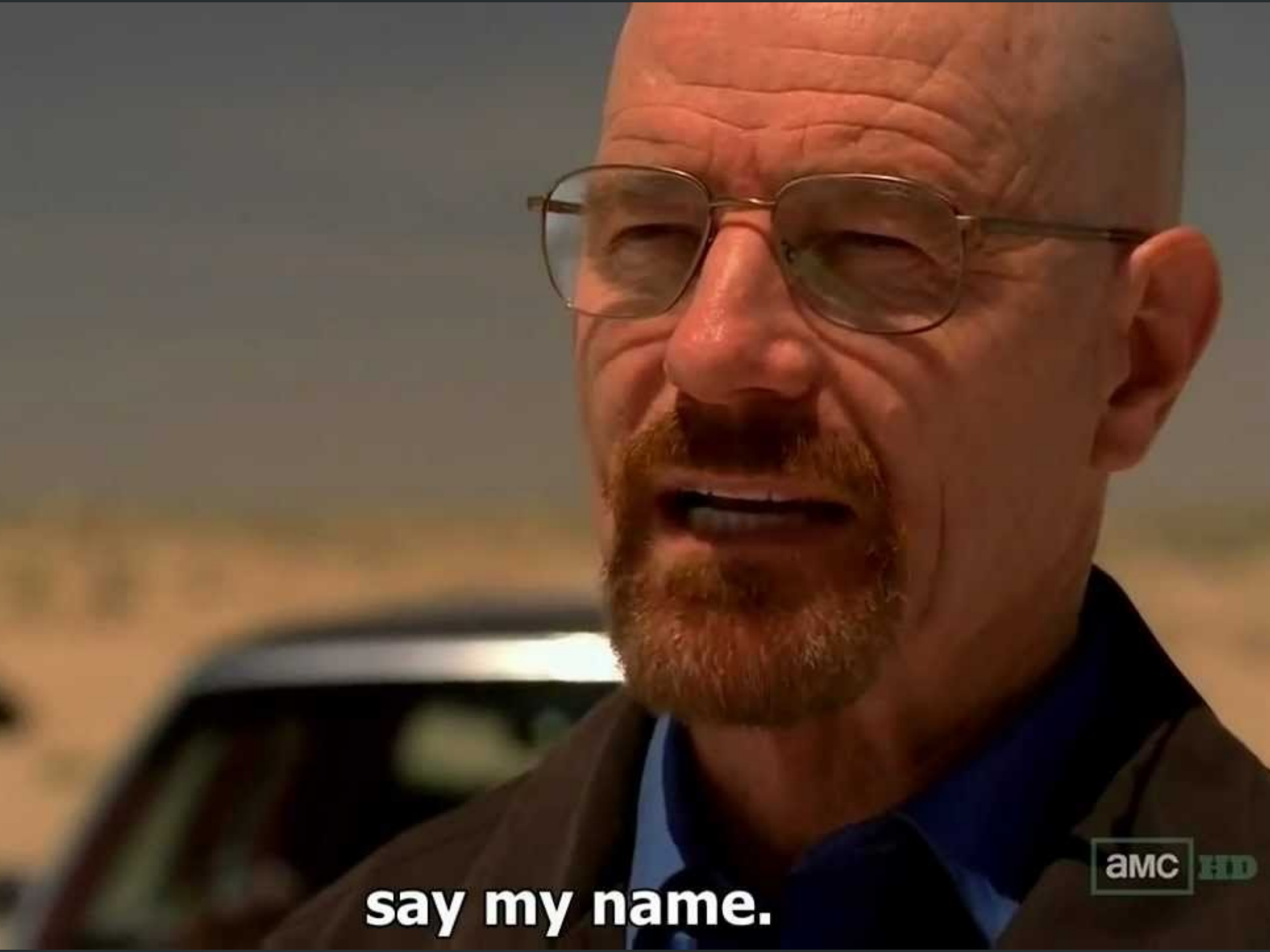
(and dynamic templating in general)

```
div.list-wrapper id="#{is_inbound ? 'inbound' : 'outbound'}"
  - way = is_inbound ? 'inbound' : 'outbound'
  = render 'shared/book_entries/import_modal', way: way if can?(:create, BookEntry)

  .form-inline.row-fluid
    = search_form_for @q, :url => target, :html => { :class => "inline input-filer" } do |f|
      = hidden_field_tag :book_entries_sort, @order_by
      = hidden_field_tag :book_entries_direction, @direction

      .row-fluid.header.page-header
        h1.pull-left
          = t(is_inbound ? 'inbound' : 'outbound', :scope => 'common.invoicing')
        hr
      .row-fluid.page_subheader
        .calendar.span5
          = render 'shared/book_entries/datepicker', is_inbound: is_inbound, target: target
        .span7#sum_header
          = render "shared/book_entries/sum", is_inbound: is_inbound
      = render 'shared/book_entries/graphics', is_inbound: is_inbound
      = render "shared/book_entries/filters", is_inbound: is_inbound, way: way

  .total_results
    span=' @total_count
    = pluralize(@total_count, t('.element')).gsub(/#{@total_count} /, '')

  - if book_entries.any?
    .legend-filter
      ul
        - unless is_inbound
          li.budget
            = link_to search_path_for_type(:budgets) do
              .filter-square
              = t('budget', scope: 'common.status')
        - %w{ payment_pending paid unpaid }.each do |state|
          li class=state
            - state_for_translation = state
            - state_for_translation = 'charged' if !is_inbound && state == 'paid'
            = link_to search_path_for_status(is_inbound, state) do
              .filter-square
              = t(state_for_translation, scope: 'common.status')

        - if is_inbound && owner.managed?
          - { verified: 'icon-ok', refused: 'icon-remove' }.each do |status, clazz|
            li
              i class=clazz
              = link_to t(status, scope: 'common.status'), search_path_for_status(true, status)

    .table-list
      table.table#book-entries-table
        - if book_entries.any?
          = render 'shared/book_entries/headers', is_inbound: is_inbound
          tbody
            = render :partial => "shared/book_entries/tr", :collection => book_entries, :as => :book_entry,
              :locals => { :lister => lister,  :is_inbound => is_inbound }, :inline => true
    = paginate @paginated_book_entries
```

Now tell me about it

- A way of creating HTML files dynamically
- Variables, arrays, conditionals… you name it
- Use composition to rock it hard

So let's improve it


say my name.
amc HD

The .rb file

```ruby
require 'sinatra'
require 'sinatra/reloader'

set :port, 3000
set :bind, '0.0.0.0'

get '/say-my-name' do
  @name = 'Walter'
  erb :index
end
```

And the .erb file

```
<html>
  <body>
    I said the name and is...
    <%= @name %>
  </body>
</html>
```

Curl it!

```
 curl localhost:3000/say-my-name
<html>
  <body>
    I said the name and is...
    Walter
  </body>
</html>
```


You're goddamn right.

LIVE CODING FROM BCN!!!!111

Let's template some stuff

IRON HACK

LET'S GET READY TO RUMBLE

Exercises SL3 & SL4