Sinatra Part Two

Query Strings

A query string is data sent in the URL using the following syntax:

http://someurl.com?queryvariablename=queryvariablevalue

- To add more query variables to the query string, we can use the & operator:

http://someurl.com?
queryvariablename=queryvariablevalue&other=value

Real world example

Google Maps uses query strings to allow you to easily link to a search, try clicking the below:

https://www.google.com/maps?q=the+new+york+code+design+academy+new+york+new+york

Sinatra: The `params` hash

- The `params` hash is a Ruby hash Sinatra used to store any data incoming to the route
- This can include submitted form data or query strings

Exercise: The `params` hash

- Try this code:

```
get '/sup' do
  puts "THESE ARE MY PARAMS"
  puts params.inspect
end
```

- Then try hitting the url in the browser while Sinatra is running /sup?hi=you and looking in the terminal at the Sinatra server logs
- Use CMD + F to search for THESE ARE MY PARAMS

Sinatra: The `params` hash

- The parameters inside the `params` hash are accessed just like any other Ruby hash, using the [] syntax:

```
puts params[:hi]
> "you"
```

Sinatra: Instance Variables

An instance variable as it pertains to Sinatra is just a variable set in a route which you can use in a view

```
#app.rb
get '/' do
  @user = User.find(1)
end

#home.erb
User email: <%= @user.email %>
```

Sinatra: Instance Variables

You could set an instance variable equal to something that comes in as a parameter in the params hash to use it in a view:

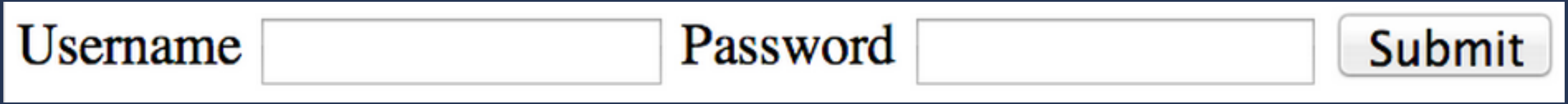
```
#app.rb
get '/' do
  @q = params[:q]
end

#home.erb
Your query was: <%= @q %>
```

Forms

- In order to accept user input, you'll need to learn how to use the HTML `<form>` element!
- `<form>` has two very important attributes
    - `method` - Used to specify which HTTP verb to use, GET or POST for this request
    - `action` - What URL should the form data be submitted to

Example Form

```
<form method="POST" action="/sign-in">
  <label for="username">Username</label>
  <input type="text" name="username">
  <label for="password">Password</label>
  <input type="password" name="password">
  <input type="submit">
</form>
```



Processing form data
Part 1

- In order to process your form being POST-ed to “/sign-in”, which is what happens when you hit the submit button, you’ll need a route in your Sinatra app
- Start by outputting the params hash to see what you have to work with

```ruby
post '/sign-in' do
  puts "my params are" + params.inspect
end
```

Processing form data
Part 2

- Now submit your form
- Inside of your terminal window you should see something like this when you submit your form:

```
127.0.0.1 - - [28/Jan/2015 12:26:01] "GET /sign-in HTTP/1.1" 200 200 0.0017
my params are{"username"=>"zach@nycda.com", "password" => "password"}
127.0.0.1 - - [28/Jan/2015 12:26:10] "POST /sign-in HTTP/1.1" 200 200 0.0017
```

- This gives you the data inside the params hash if you ever need it for debugging purposes

Exercise

- Create a simple "Contact Us" form
- It should use a `<textarea>` input element for the message portion of the form
- Try submitting the form to a POST route in your Sinatra app and see if you can recognize the input in the terminal window