cookies & sessions

Daniel Jackson
HTTP is stateless

why no session state?
› so memory doesn’t grow with number of clients
› if client dies, no problem for server
› also, good for load balancing amongst servers

but we need session state!
› eg, whether logged in, content of shopping cart

solution
› server sends state to client as ‘cookie’
› client sends it back on each request

often
› server stores some session state, sends session id in cookie
cookies in http

cookie is
› name-value pair
› expiration, path & domain

server sends
› using set-cookie header

browser sends back
› all unexpired cookies
› with matching path expiration

expiration
› session cookies: on quit
› persistent cookies: on expire

a funny cookie tale
nytimes.com used cookies to count #articles read, so viewers just deleted cookies...
example use of cookies: logging in

step 1: user opens home page
› request includes no cookies
› response body includes no member content

step 2: user submits login form
› request is POST with user and password as data
› response includes set-cookie headers <user: dnj, login: true>

step 3: user reopens home page
› request includes all cookies for domain
› response body includes member content

how to prevent cookies being faked?
› server encrypts cookie values with secret key
session state

```ruby
# save state in session
session[:user_id] = @current_user.id

# get state from session
User.find(session[:user_id])
```

session variable holds session state

› Rails magic: session is not a regular hash
› get and set methods on session[] access the cookies

where’s session state stored?

› by default, just in browser cookies

store big shopping cart in session?

› if user clears cookies or closes browser, all lost
› cookies limited to 4kB
authenticating the user

```ruby
class LoginsController < ApplicationController
  def create
    if user = User.authenticate(params[:username], params[:password])
      # Save the user ID in the session so it can be used in
      # subsequent requests
      session[:current_user_id] = user.id
      redirect_to root_url
    end
  end
end
```

note that check is in model class

› separation of concerns
› keep all data-related logic in models
class LoginsController < ApplicationController
  def destroy
    session[:current_user_id] = nil
    flash[:notice] = "You have successfully logged out."
    redirect_to root_url
  end
end

what is flash?
› like session, but only lasts one request
› message will be displayed in response to next request
getting the current user

class ApplicationController < ActionController::Base

private
  def current_user
    @_current_user ||= User.find_by_id(session[:current_user_id])
  end
end

what does this do?
› if @_current_user is defined, returns it
› else looks up in database, using user id from session
controlling access

class ApplicationController < ActionController::Base
  before_action :require_login

  private

  def require_login
    unless current_user
      flash[:error] = "You must be logged in to access this section"
      redirect_to new_login_url # halts request cycle
    end
  end
end

class LoginsController < ApplicationController
  skip_before_action :require_login, only: [:new, :create]
end

before action will prevent action if not logged in
  but must still check that user is the right one!