Lesson 14:

Cookies, sessions & authentication

Authenticating and managing users

– Understand sessions and cookies
– Understand authentication
– Store identities and user data
– Implement basic authentication
– Authentication gems

What is a session?

By default, a server stores no data between your browser's requests.

A ***session*** is a set of browser HTTP requests for which some data is stored between requests.

Session data commonly includes the user's identity (name, login, shopping cart, etc.)

The web server must uniquely identify each browser's particular HTTP requests while the session lasts. Commonly, servers identify browsers by asking them to store a cookie.

What is a cookie?

A ***cookie*** is a small text file offered to a browser by a server using a HTTP response header

Browsers send all cookies from a domain back on each request.

Cookies are stored in the browser along with the setting domain, timestamp, and intended duration.

Browser storage is ***not*** secure and sensitive data (password, credit cards, etc.) ***should not*** be set in a cookie.

Cookies allow web servers to provide a temporary unique ID to a browser, to enable session management.

How does Rails set and access cookies?

Rails creates a cookies hash for each page request, commonly accessed in the controller.

Values in the cookies hash are passed to and from browser.

How does Rails set and access cookies?

Three places data may be stored in Rails:

– ***Variables*** in memory
– ***Parameters*** sent by browser
– ***Cookies*** in the cookies hash, sent by browser and returned by web server

```
def login
    @previous_memory_name = cookies[:stored_name]
    @memory_name = params[:input_name]
    cookies[:stored_name] = @memory_name
end
```

Exercise:

Setting & accessing cookies

How can cookies be configured?

Cookies may be directly set as a simple value

```
cookies[:name] = “Fred”
```

Cookies may be assigned values to configure their behavior

```
cookies[:name] = {
    value: “Fred”,
    secure: true,
    expires: 5.minutes
}
```

How can cookies be configured?

:value	a short string, commonly a db key for a user record
:domain	domain or subdomain serving the cookie
:path	the path of the file serving the cookie, or part of the path
:expires	how long the cookie will remain
:secure	if true, cookie will only be sent over HTTPS connections
:http_only	if true, will be sent with HTTP/S requests, but not be available to Javascript on the page

What is session management?

While values can be kept cookies, it can be cumbersome and insecure

Rails session management:
– tracks the user during the ***session*** with a single cookie
– allows you to store any number of values in memory, keyed to that user's session cookie

The `session` object:
– is available to the controller and view
– can store any number of values in memory for duration of the current user's session

How sessions work?

Session may be directly set as a simple value

```
session[:name] = “Fred”
```

Exercise:

Using session data

What is authentication?

Authentication is verifying that someone or something is who they claim to be

Authorization is allowing an authenticated user the privileges needed to use a system

What is involved in an authentication process?

1. User requests a page looking for a session variable
2. If no session variable exists, user is directed to a login screen
3. User inputs credentials (e.g. username and password)
4. App checks whether the credentials are known and valid
5. App recalls user data associated with these credentials from the model
6. App initiates a new user session variable with that user's information
7. User is redirected to an initial application screen
8. User now passes the security filter for each page due to the session variable

How does rails 3 implement simple authentication?

There are numerous ways to implement authentication.

Rails 3 provides a new, streamlined mechanism using ActiveModel called has_secure_password

The BCrypt gem is used to provide ***encryption*** support for passwords and BCrypt comes pre-installed with Rails 3 but must be configured in the Gemfile

```
# To use ActiveModel has_secure_password
gem 'bcrypt-ruby', '~> 3.0.0'
```

Exercise:

Implementing Rails 3 authentication

What other authentication frameworks are available?

Numerous Rails authentication frameworks are available which may extend simple authentication, with configurable user data management and support for various back end authentication sources (Twitter, Facebook, etc.)

OmniAuth:
http://www.omniauth.org/

Devise:
https://github.com/plataformatec/devise