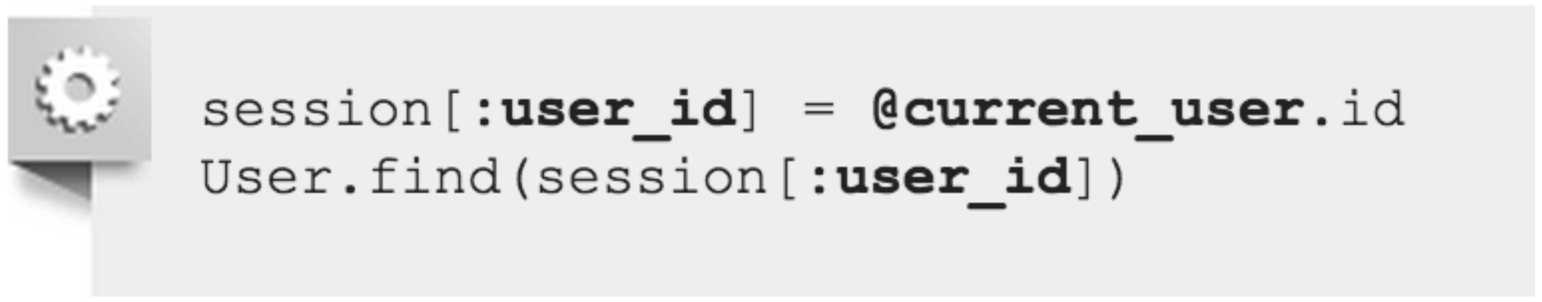
User Data

Session

Data persisted across page loads using the cookie.
Rails gives this to you automagically.

Accessing a Session

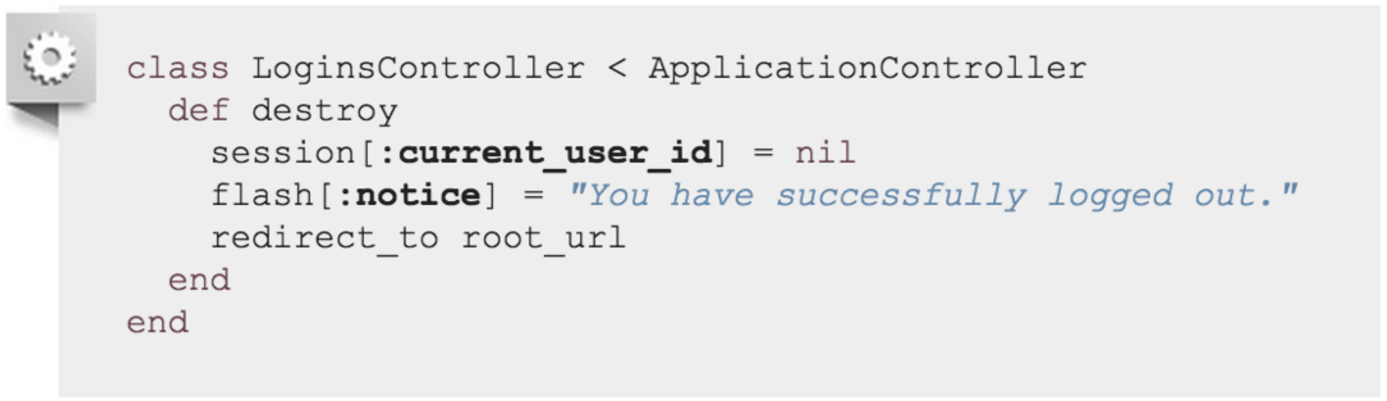


```
session[:user_id] = @current_user.id
User.find(session[:user_id])
```

http://guides.rubyonrails.org/action_controller_overview.html#session

Flash

session data cleared at the end of the next request



```ruby
class LoginsController < ApplicationController
  def destroy
    session[:current_user_id] = nil
    flash[:notice] = "You have successfully logged out."
    redirect_to root_url
  end
end
```

http://guides.rubyonrails.org/action_controller_overview.html#session

CookieStore

default session storage mechanism.

all session data is stored in the user's cookie.

cookies have a 4k limit.

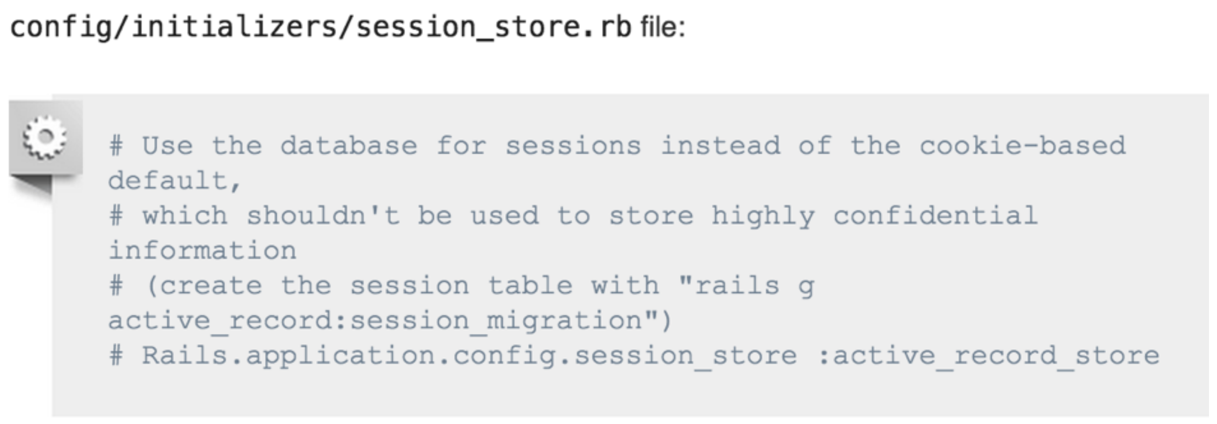
Too much Data?

add a user_id to the session and
access the DB data with it

OR

ActiveRecordStore

stores session data in the database.



`config/initializers/session_store.rb` file:

```
# Use the database for sessions instead of the cookie-based
default,
# which shouldn't be used to store highly confidential
information
# (create the session table with "rails g
active_record:session_migration")
# Rails.application.config.session_store :active_record_store
```

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

Session ID

md5 hash of a random string. the string has four parts:

<current time><random num between 0 and 1>
<pid of the interpreter><constant string>

sent with every request (through the cookie)

Security

Session Hijacking

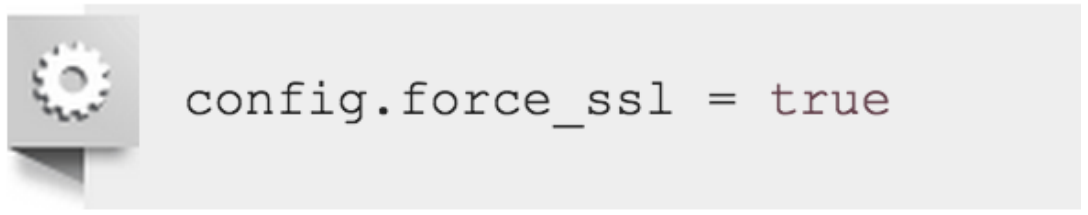
unauthorized access through using someone else's session (by stealing their cookie)

Sniffing

listening to network traffic

SSL

encryption between the browser and website



```
config.force_ssl = true
```

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

Replay Attacks

Replacing the current cookie with a previous one.

Counter Measure

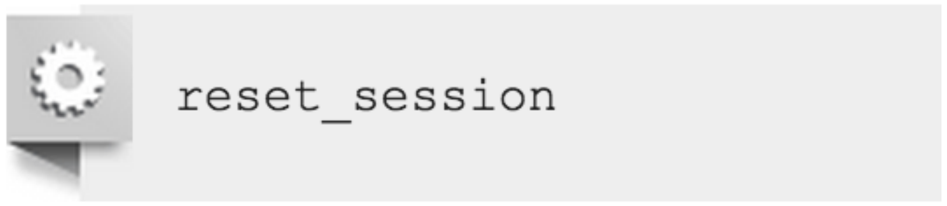
don't store things like store credit in the cookie.

Session Fixation

attacker creates a valid session and then forces the victim's browser to use that session id.

Counter Measure

reset the session after each login.



```
reset_session
```

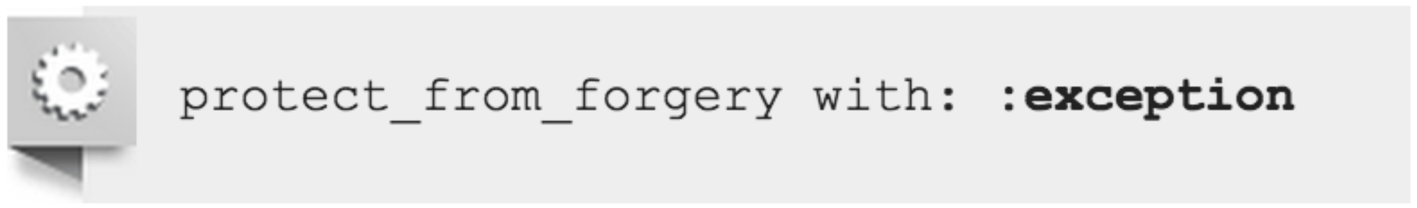
Cross Site Request Forgery

hacker injects a get/post request using an html (img, link) tag

Counter Measures

make sure get requests are idempotent

make sure authentication tokens are required for posts

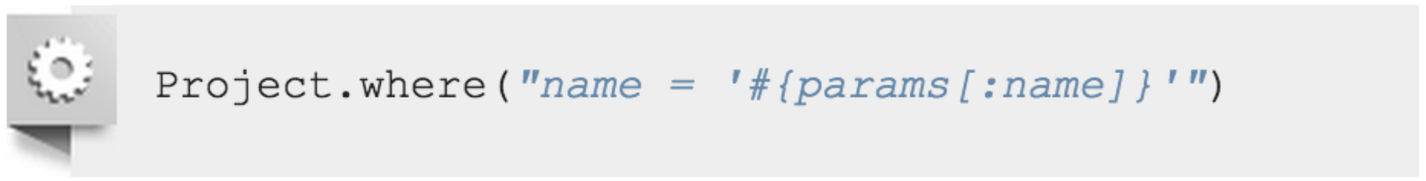


```
protect_from_forgery with: :exception
```

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

Sql Injection

including fragments of a sql query in user input fields



```
Project.where("name = '#{params[:name]}'")
```

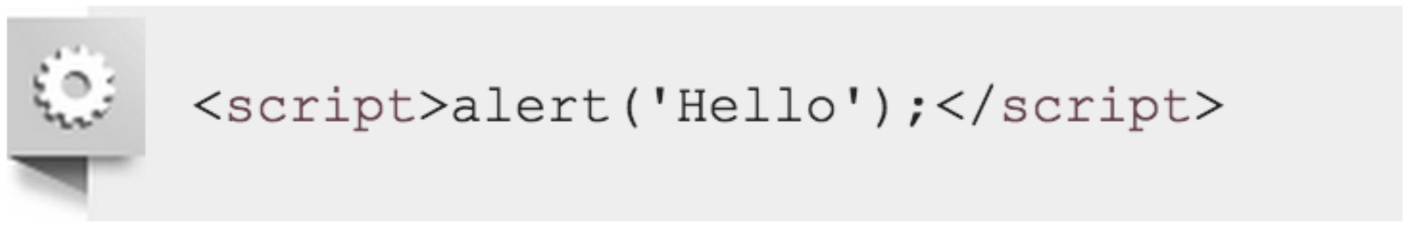
http://guides.rubyonrails.org/security.html

Counter Measures

don't use user input in sequel queries

Cross Site Scripting (XSS)

Adding javascript to user input fields (where the content is displayed to other users).



```
<script>alert('Hello');</script>
```

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

Redirection

redirect a user to a site that looks/acts the same.

```
def legacy
  redirect_to(params.update(action:'main'))
end
```

```
http://www.example.com/site/legacy?
param1=xy&param2=23&host=www.attacker.com
```

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

Counter Measures

don't include user data in a redirect

Authentication

verify that requests are coming from a specific user

Use Devise

http://guides.railsgirls.com/devise/