Authentication

From Scratch

Authentication

Objectives

1. Provide users with a way to login to your app
2. Store password encrypted in database
3. Create sessions so users stay logged in
4. Destroy session to logout user

Encryption

Storing a user’s password in plain text in your database is a big TO DON’T

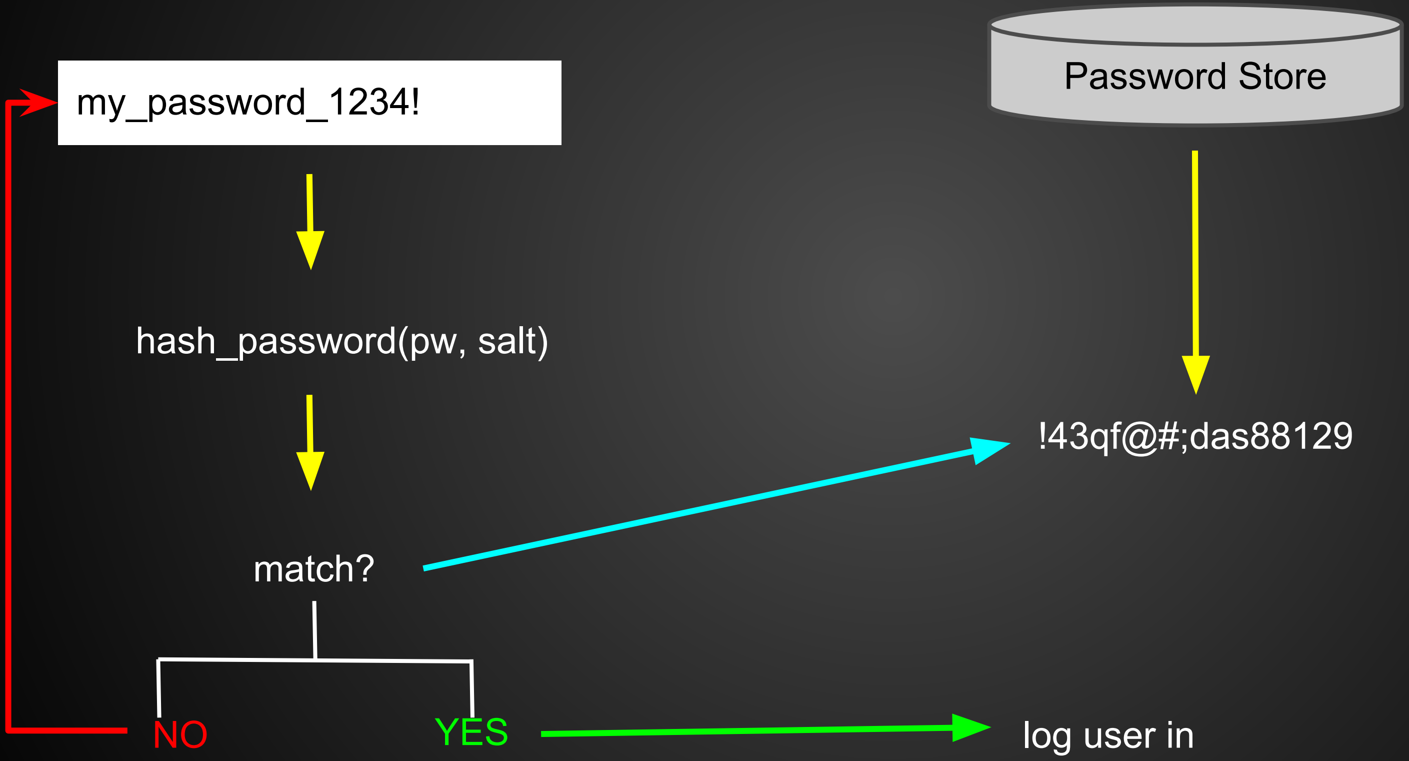
We need to be able to encrypt passwords so that the password is secure in the database.

How encryption works:

Encryption typically takes in the password and a “salt” and returns a hashed password. Then instead of checking if the password matches in the database it checks if the password_hash matches the text passed in and hashed.

MORE SECURE!

Encryption (cont)



Build an app

Create new rails app

```
rails new first_bank -d postgresql --quiet
cd first_bank
```

Create users controller

```
rails g controller users new
rails g controller home index
```

NOTICE: We are not storing a password

Create users model

```
rails g model User email:string password_digest:string
```

Create and migrate database

```
bundle exec rake db:create db:migrate
```

Update config/routes.rb

```
Rails.application.routes.draw do
  root ‘home#index’
  get ‘/signup’, to: ‘users#new’, as: ‘signup’
  post ‘/signup’, to: ‘users#create’
  resources :users
  resources :home
end
```

Set the root to home

Change the get signup route to be more friendly

Add resources for the user

Add resources for home

Views

app/views/home/index.html.erb

```
<h2>Welcome To First Bank</h2>
<%= link_to 'Sign Up', signup_path %>
```

app/views/users/new.html.erb

```
<h2>Sign Up</h2>
<%= form_for :user do |f| %>
  <%= f.label :email %>
  <%= f.text_field :email %>
  <%= f.label :password %>
  <%= f.password_field :password %>
  <%= f.label :password_confirmation %>
  <%= f.password_field :password_confirmation %>
  <%= f.submit %>
<% end %>
```

NOTICE: There is a password attribute in the form but users do not have a password attribute. For passwords use f.password_field which will create: <input type="password">

app/controllers/users_controller.rb

```
class UsersController < ApplicationController
  def new
  end


  def create
   user = User.new(user_params)
    if user.save
      render text: “User #{user.email} created”
    else
      render :new
    end
  end


  private
  def user_params
    params.require(:user).permit(:email, :password, :password_confirmation)
  end
end
```

For now the app will just render text so that it’s possible to see if it actually worked. Later we will redirect to a route

Why does this work? User does not have password or password confirmation.

app/models/users.rb

```ruby
class User < ActiveRecord::Base
  validates_presence_of :email, :password_digest
  has_secure_password
end
```

Gemfile.rb

```ruby
gem ‘bcrypt’
```

```
> bundle
```

The bcrypt gem will do all of the password hashing behind the scenes.

This is why you can pass password and password confirmation as user params and also gives the has_secure_password method used above.

The next step is to create a session controller to provide existing users with a way to login and log out.

```
rails g controller sessions new
```

app/controllers/sessions_controller.rb

```
class SessionsController < ActiveRecord::Base

  def new
  end

  def create
  end

  def destroy
  end
end
```

Stub out the create and destroy methods.

app/controllers/sessions_controller.rb

```
def create
  user = User.find_by_email(params[:email])
  if user && user.authenticate(params[:password])
    session[:user_id] = user.id
    render text: ‘User signed in’
  else
    redirect_to ‘/login’
  end
end
```

user.authenticate will take the password and use the same hash salt to see if it produces the same hash as the user's password hash

When a session is created a session is set in the browser cookies

app/controllers/sessions_controller.rb

```
def destroy
  session[:user_id] = nil
  redirect_to ‘/’
end
```

When a session is destroyed the browser cookie is wiped out app/views/sessions/new.html.erb

```
<h2>Login</h2>

<%= form_tag '/login' do %>
  Email: <%= text_field_tag :email %>
  Password: <%= password_field_tag :password %>
  <%= submit_tag "Submit" %>
<% end %>
```

app/views/home/index.html.erb

```
<h2>Welcome To First Bank</h2>
<%= link_to 'Sign Up', signup_path %>
<%= link_to 'Login', login_path %>
```

config/routes.rb

```
Rails.application.routes.draw do
  root 'users#new'
  get '/signup', to: 'users#new', as: 'signup'
  get '/login', to: 'sessions#new', as: 'login'
  get '/logout', to: 'sessions#destroy', as: 'logout'

  post '/signup', to: 'users#create'
  post '/login', to: 'sessions#create'
  resources :users
  resources :home
end
```

If you don't have a model to work with you can still create a form using form_tag instead of f.text_field you can use text_field_tag app/controllers/application_controller.rb

```
class ApplicationController < ActionController::Base
  protect_from_forgery with: :exception


   def current_user
    @current_user ||= User.find(session[:user_id]) if session[:user_id]
  end


  def authorize
    redirect_to '/login' unless current_user
  end
end
```

Shared methods that need to be used app wide can be located in ApplicationController since all other controllers are children of ApplicationController

```
rails g controller Accounts show
```

app/controllers/accounts_controller.rb

```ruby
class AccountsController < ApplicationController
  before_action :authorize

  def show
    @user = current_user
  end
end
```

Remove the accounts get from config/routes.rb and add resources :accounts

app/views/accounts/show.html.erb

```erb
<%= link_to ‘Logout’,  logout_path(@user) %>
<h2>Welcome <%= @user.email %></h2>
```

app/controllers/sessions_controller.rb

```ruby
 def create
   user = User.find_by_email(params[:email])
   if user && user.authenticate(params[:password])
     session[:user_id] = user.id
     redirect_to account_path(current_user)
   else
     redirect_to ‘/login’
   end
 end
```

app/controllers/users_controller.rb

```
def create
  user = User.new(user_params)
  if user.save
    session[:user_id] = user.id
    redirect_to account_path(current_user)
  else
    new
  end
end
```

Set the session[:user_id] when creating a user so that they are logged in and stay logged in