Take Home Skill Check November

skill_check_nov.md Raw

Question Set 1

1. c
2. The wording is unclear here. It says creating an existing record. If you mean updating an existing record, the answer would be B. update. If you mean creating a new record, the answer would be A. put
3. a

Question Set 2

1. The browser initiates a request of a webpage, which is sent to the web server. The server sends the URL to the router, which maps it to the http request verb, controller and action (e.g. POST with the users controller and update action). The controller might also save some things in the URL to the params hash variable, such as the :id of the user to be updated. The controller executes the method corresponding to the action identified by the router. The method might query the model in order to store the relevant information to send to the view. For example, the update method might query the User table in the model and save it in an instance variable (@user). The controller then uses the information in the model to render the html in the view. This html is then sent back to the client in the response. The benefit of the Model-View-Controller architecture is the separation of concerns. Models and Views do not "know" about each other and therefore it is easy to redesign the UI or restructure the database without breaking the application. It helps prevent spaghetti code.

2.

- Create: GET the form, then POST
- Read: GET
- Update: GET the form, then PATCH
- Delete: DELETE

Question Set 3

1. I'm not great with this syntax. I'm sure I'm wrong. I'd have to look it up. I'm also not clear on what the question is asking in terms of the one text field. Is it an input? Also can't remember the submit syntax.

```
<%= form_for @user %>
  <% text_field_tag body %>
  <% submit_button %>
<% end %>
```

1. Using `redirect_to` is the same as a client sending a request for the given page. For example, after creating a new user the controller often will redirect to the user's information page. This is the same as if the client had typed in the URL for accessing the user page (i.e. /users/:id). Using `render` tells the controller to load a certain html.erb file into the fully rendered html. One example of for using render is when including partials of headers, footers, or error messages in the view. Another example is when a user submits a form that is only partially complete. The controller can display an error message and then re-render the form.

Question Set 4

1.

```
class PhotosController < Rails::ControllerBase # clearly guessing here
  def index
    @photos = Photo.all
  end

  def new
    @photo = Photo.new()
  end
end
```

2.

a. Photo.find(430)

b. Photo.last

c. Photo.where('id > 99 AND id < 106')

3.

```
get "/photos" => "photos#index"
get "/photos/new" => "photos#new"
```

get = HTTP verb

"/photos" and "photos/new" = URL

photos = controller

index and new = action

4.

```
<h1> Display All Photos</h1>
<p>
<%= @photos.each do |photo| %>
  <img url="<%= photo.url %>" alt="<%= photo.description %>">
  <caption><%= photo.description %></caption>
<% end %>
</p>
```

Question Set 5

1.

```
Post.since(Time.now)
```

2.

```
post = Post.find(17)
post.comments.good.count
```

3.

```
class Post < ActiveRecord::Base
  def most_good_comments
    posts = Post.all
    posts.max_by { |post| post.comments.good.count }
  end
end
```

\4. score and post_id

5. a. post.comment = comment b. comment.post_id = 4