Josh Dinges

Web Apps Midterm

March 11, 2012

**Architecture**

1. What is the difference between an HTTP PUT request and a POST request?

The URI in a POST request will identify the resource that will handle its entity. In contrast, the PUT request can only identify the entity enclosed, and NOT the resource it will use; the user agent already knows this resource.

You see a link in an interface whose markup is as follows:

<a href="images/new?request\_type=PUT" method="POST">create a new image</a>

1. Is the target URL relative or absolute?

It’s relative.

1. What is the difference between an absolute and relative URL?

The difference between a relative and absolute URL is that an absolute URL will identify the file path to the document, the server it’s located on, and the protocol necessary to access the document. A relative URL, on the other hand, will assume that the server remembers the file path and protocol to your current location and you need only specify the file path to the document you wish to reference. The specification is as follows: you can access any files in your current directory, if you wish to access files outside your current directory, simply specify the first common parent directory between your location and the location of your target file.

1. If you clicked on this link, what kind of request would your browser generate? (which HTTP method?) Assume no JavaScript modifies the behavior of the link.

POST

1. Is there a querystring? What is it?

Yes, it’s the request\_type=PUT

1. What is lacking from the link declaration that would otherwise enhance accessibility?

You could add an alt attribute.

1. What are the roles of the database and Web browser in most Web applications? (One sentence for each.)

The web browser should display and translate HTML, allow users to follow links, and manage downloads. The database should store all of the documents, images, etc. for the user.

Given the following HTTP response header:

HTTP/1.1 200 OK  
Date: Wed, 09 Mar 2011 16:43:33 GMT  
Server: Apache  
Connection: Keep-Alive  
Keep-Alive: timeout=2, max=100  
Etag: "110e412f-7df-49e0f6a106500"  
Vary: Accept-Encoding

1. Would an HTTP response that begins like that usually contain a body? Why or why not?

That header would not usually contain a body because it lacks lines specifying the content type and the content length.

**Ruby**

1. Write a Ruby class definition that meets the following criteria:
   * class is called Troll
   * class has publicly accessible attributes ugliness, smelliness, and strength
   * upon instantiation, an object of this class has a member variable, a String, called grunt, whose initial value is "UNGAH" (that's pronounce "oon-guh").
   * class has an instance method called speak() that prints the value of the instance variable grunt 42 times
   * class has an instance method called reverse() that prints the value of the instance variable grunt backwards
   * class has a static/class method called propagate(), which returns a Troll instance whose grunt attribute is "eegah"
2. Imagine a Troll instance fred, which, when the following method is called:

fred.respond\_to?("fight")

returns true. What is missing from your class definition in order for this example to be accurate?

def respond\_to?(action)

#Some code that makes function work

end

1. Does the respond\_to?() method illustrate object-oriented polymorphism? If so, in what manner?

No, because that function name is not used anywhere else to accomplish a similar goal.

1. According to Ruby conventions, what kind of value would you expect to receive from a method that ends in a question mark (?) ?

Boolean

1. According to Ruby conventions, what is the difference between pairs of methods like do\_this and do\_this! (notice the bang) ?

The do\_this method would return something (probably an object) that has been modified according to the method definition, that is, it is the object after the method has been run on itself. On the other hand, do\_this! would not return anything, but modify the object itself.

1. Briefly explain Ruby's type system. What is it (by name)? What does it mean?

Ruby is a dynamically typed system. What that means is that variables do not have to be declared to be of a given type. For example, in Java you would have had to declare

int x;

Where ‘int’ tells Java that ‘x’ is going to store an integer value. Ruby does not require this definition. It will look at what’s going into ‘x’ and determine how to save that value all on its own. It also means that you can change what goes into x as you wish. For example:

x = 42

x = “Real big fish”

This is perfectly valid. At first, you’ll notice, ‘x’ is storing the value (presumably an integer) 42, but then on the next line it’s storing a string value; this is legal behavior in a dynamically typed language such as Ruby.

1. What type of Ruby object does the following expression yield? %w( master rails and then try another framework you'll never go back)

It yields an Array object.

1. Given an array of strings called @happy\_places, would these two snippets of code do the same thing?

@happy\_places.each do |happy\_place|  
  puts happy\_place  
end

and

@happy\_places.each {|hp| puts hp}

Yes. They are both valid.

1. Given a function that needs to return a value to its caller, does the function need an explicit return statement? If so, explain why. If not, then what can you always expect a Ruby function to return?

False. In the Ruby language, the result of the last line in a function is what is returned to the caller. You can always expect a Ruby function to return the result of last line of code in a function executed.

**Rails**

1. Name four ActiveRecord callbacks that you can bind methods to.

Method reference (a symbol), callback objects, inline methods (using proc), inline eval methods (string)

1. The Rails convention maps HTTP methods to certain controller methods, and those methods usually involve specific CRUD operations on models. Given the following CRUD database methods:  
   create, read, update, and delete  
   and the following HTTP methods:  
   GET, PUT, POST, DELETE  
   and the following controller actions:  
   index, new, create, edit, update, destroy Complete the following table.

|  |  |  |
| --- | --- | --- |
| **HTTP method** | **controller action** | **CRUD operation** |
| get | index | read |
| get | new | create |
| post | create | create |
| get | edit | update |
| Put | update | update |
| delete | destroy | delete |

1. Rails "simulates" PUT and DELETE requests. Why?

Most browsers don’t support the PUT and DELETE requests for forms. Most browsers only support GET and POST.

1. What is the difference between the two Rails environments 'production' and 'development' ?

The rails dispatcher, in development mode, will automatically reload source files so that any changes we make will show up right away without having to restart the application. This comes at a cost, namely speed. It takes a little bit after a URL has been entered to load the page. This dip in performance is unacceptable at the production level and is the reason why it’s disabled.

1. Usually, Rails controllers incorporate plural nouns, such as ProtestsController and RevolutionsController. In what case should a controller have a singular name like GeocodingController?

When that controller controls a single resource and not a collection of resources. In the example above, GeocodingController controls a single geocoding resource, not a collection of them.

1. What is a Rails "helper method" and when should they be defined and used by you, the developer?

A helper method is a method that is used by a controller to help reduce repeated code. Usually we need to use helpers when dealing with the view. Helpers should be used in the place of repeated code in the view.

Assume you have a Flower AR class that has\_and\_belongs\_to\_many :bees, and a Bee class that has\_and\_belongs\_to\_many :flowers.

1. What must exist in the database schema in order for AR to infer the proper foreign key / relationship?

There needs to be a join table for the two entities. For example, you could have a bees\_to\_flowers table that just has two columns: bee\_id and flower\_id.

Assume that a Bee :belongs\_to a Hive and a Hive has\_many :bees. Also assume a GET request is sent to the FlowersController#show action, which contains a finder method call @flower = Flower.find(params[:id]. Assume the view app/views/flowers/show.html.haml displays the name of the Flower and each Flower's bee's name and Hive name like so:

- @flower.bees.each do |b|  
  %h1= b.name  
  %p= b.hive.name

If you were tailing the log of your application during the rendering of the response, you would notice tons of database queries.

1. Are all of those queries ok? If so, explain why. If not, explain how you would reduce the number of database queries (without hand-rolling your own SQL query).

This many SQL queries is generally a bad idea. The reason that this is troublesome is that it puts unnecessary pressure on the database to constantly respond to all of those queries. We may not notice it in our small application, but if it were rolled out production and our site started getting a lot of users, this performance slow down would be noticeable. The way to fix that would be to modify your @flower method in the FlowersController. It should read:

@flower = Flower.find(params[:id], :include => :bees)

This illustrates the idea of eager loading. Where we load what we need from the database and store it locally instead of having to constantly bombard the database with SQL queries.