Cecilia Smyth

CSCI 446

Midterm Exam

1. The difference between an HTTP PUT request and a POST request is a matter of handling and URIs; a PUT request has an intended URI and cannot be applied to a different URI without giving a 301 (Moved Permanently) response. On the other hand, the URI of a POST request identifies a resource to be used in handling the request, though what this resource is may vary.
2. The target URL is relative because it does not contain a full path name but instead refers to a file in the current directory.
3. An absolute URL contains the full path to the named file/resource whereas a relative URL contains a partial path, depending on the relation to the current file/directory. Absolute URLs are generally longer and more cumbersome and can point to resources on a different server than the current one, but are also less portable.
4. If you clicked on this link, the browser would generate a POST request.
5. Yes. The querystring is “request\_type=PUT”
6. Including the title attribute in the link declaration would enhance accessibility.
7. The database and Web browser in Web applications are not unlike the controller and view in an MVC architecture, respectively. The browser displays the application. The database stores information about the application. Requests are sent to the database in order to update or perform actions from the browser.
8. An HTTP response that begins like this would usually contain a body, as indicated by the HTTP status code “200.” Codes of the form 2xx indicate the successful processing of a request and that the resource requested is contained therein.

class Troll

attr\_accessor :ugliness, :smelliness, :strength

@grunt = "UNGAH"

def speak()

for i in 1..42

puts @grunt

end

end

def reverse()

puts @grunt.reverse()

end

def self.propagate()

temp = Troll.new()

temp.grunt = "eegah"

return temp

end

end

1. The instance method called fight() is the missing piece that would cause fred.respond\_to?(“fight”) to return true.
2. Yes, the respond\_to?() method illustrates object-oriented polymorphism because it can be used with any object, no matter the object’s specifications.
3. I expect to receive a Boolean (true or false) value from a method that ends in a question mark.
4. According to Ruby conventions, methods that end in exclamation points (such as do\_this!) are generally thought of as “dangerous” methods because they could potentially change the calling object, arguments, or self, whereas methods that do not end with a bang are considered “safe” because, if they modify something, they only modify a copy and not the original.
5. Ruby is dynamically typed, meaning that data types for all included objects are determined (via type checking) at run-time. In a statically typed language, type checking is done when the code is compiled.
6. The expression yields an array of words.
7. The two snippets of code would do the same thing.
8. In Ruby, functions do not need explicit return statements because the value of the last statement in the function (generally, the line before end) is automatically returned. However, an explicit return statement can still be used, if so desired.
9. Four ActiveRecord callback that you can bind methods to: before\_save, after\_save, before\_destroy, after\_destroy

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Controller Action** | **CRUD Operation** |
| GET | *index* | read |
| GET | *new* | create |
| POST | *create* | create |
| GET | *edit* | update |
| PUT | *update* | update |
| DELETE | *delete* | delete |

1. Rails “simulates” PUT and DELETE requests because some browsers only support GET and POST.