Name: Brennan K. Kimura

Date: March 9, 2012

1. POST sends data to a specific URL, whereas PUT puts a webpage at a specified URL. The data sent by POST is dealt with however the server side decides to deal with it. PUT is an idempotent operation, which means that doing the operation PUT multiple times is no different than doing it only once.
2. The target URL is relative.
3. The difference between a relative and an absolute URL is that the absolute URL contains the entire URL unlike the relative URL, which contains a partial URL.
4. The browser would generate a POST request.
5. Yes, there is a query string. The query string sets the field, request\_type, to the value, PUT.
6. The link declaration is missing a title, which the browser will use to create a tool tip.
7. The role of the database is to store information for an application in a way that is optimal for the application and its framework. The role of the browser is to display whatever html the application’s server returns to the browser.
8. This is a trick question because the response header contains the status code 200, which means that the clients request was successfully, received and understood by the server. This means that depending on the client’s request there might or might not be a body in the response header. For example a HEAD request would return nothing in the message/body of the response, a TRACE request would return the original request as received by the server in the message of the response.
9. class Troll

attr\_accessor :ugliness, :smelliness, :strength

@grunt = “UNGAH”

def speak()

42.times {puts @grunt}

end

def reverse()

puts @grunt.reverse

end

def self.propogate()

@grunt = “eegah”

end

1. If fred.respond\_to?(“fight”) returns true that means my class definition is missing a method called fight.
2. Yes, the respond\_to?() method illustrates polymorphism because it allows for different types to call it just like an overloaded operator, which is an example of polymorphism.
3. In Ruby methods ending with ‘?’ usually return with a Boolean value of true or false.
4. According to Ruby convention the ‘!’ at the end of the method means that if there is an exception thrown that it is thrown first and it makes debugging a lot easier.
5. Ruby uses a dynamic type system. The name of the dynamic type system is Duck typing. Duck typing is based on the “Duck test” and it is implemented in such a way that it doesn’t check the type of the object but whether the objects has the methods and if it doesn’t a run-time error is called.
6. It is an array of strings separated by white space.
7. They do the same thing.
8. No, Ruby will return from the function with a nil.
9. Four ActiveRecord callbacks that I can bind to methods are create, save, valid, and update.

19)

|  |  |  |
| --- | --- | --- |
| HTTP method | Controller action | CRUD operation |
|  | Index | Read |
|  | New | Read |
|  | Create | Create |
|  | Edit | Read |
|  | Update | Update |
|  | Delete | Delete |