Exploring Programming In Ruby – Practical 9

Marc Laffan (15202421)

Q1). Go back to the hello program that does the mortgage stuff and get a better idea of how the different records are being created as you go along. Have a close look att the methods in the Other Controller (esp. index, change and quote) and describe how they are interacting with the database. They could be better, no? So, modify each method to make it interact with the db in a better way.

The methods used are as follows:

- name
- index
- show
- quote
- change

name: This method creates an Entry object using the :fst_name and :lst_name parameters while simultaneously saving them to the database. It uses the create method to do both of these actions in a single line of code (Sans the creation of the @fname and @lname parameters). From here, the view displays a link containing the name parameters that sends the user to the index page. It should be noted at the point that the Entry is saved to the database, but that not all of its attributes are filled out.

index: This method finds the last Entry row in the database (Which should point to the Entry created by the name method) and updates that Entry row with the address, salary, loan amount and loan reason from the index view and saves it to the database. Following this, the method render the show page if the address used is not nil.

show: This method simply retrieves the last Entry record that was created from the database. This should be the Entry created in the name method updated in the index method. It then displays these results on a view. It should be noted that this method finds the last Entry row using the parameter ":last" with the method "find", rather than simply using the method "last".

quote: This method takes the last Entry record that was created (Again, this should be the one that was used in the show method) and firstly checks to see if the salary attribute is nill for the Entry record (if it is then assign "Seem to have an empty record??" to the @message variable). If the salary attributes is not nil however, then check to see if the loan attribute is less than three times the salary. If it is, then assign the @message variable a nasty rejection message. If it isn't, then assign @message an approving message.

change: This method takes the last Entry record that was created (Again, this should be the one that was used in the quote method) and puts that rows first_name and last_name attributes into the variables @fname and @lname. Following this, the controller creates an entirely new row that is then populated with these two attributes. The controller is run once again when the user clicks "send" and the row that was just created is recalled once more for the creation of a final Entry row that uses the name attributes from the previously created row, as well as the salary and loan parameters entered on the page.

In conclusion, I have learned that Rails controllers can be run several times on the same page based on the contents of the view, the controller and MVC itself.

Exploring Programming In Ruby – Practical 9

Q3). Finally, is it possible to move between pages without having an explicit link on which a user clicks? If so, how?

Using redirect_to, it is possible to redirect a users request to another page. This is applicable in a variety of situations, two examples being:

1. A parameter is not being sent when a user visits a particular URL. In this case, it would be similar to following the "/other/quote" URL directly, rather than starting at "/other/name" as originally intended. The original design would guarantee that the @person variable would have a value, as the database would contain at least one Entry row. However, visiting the URL directly before any entries are in the database would normally cause an issue. Redirecting the user to "/other/name" when the @person variable is nil is an alternative solution to this issue.
2. A user creating a non-existent URL. For example if the user simply typed gibberish after the domain name of the Rails application, the redirect_to could be used to point them towards the "/other/name" page. Typically this will be done in the Application Controller, along with a flash message to let the user know that they cannot perform that particular action: flash[:error] = "Error: Non-existent URL!"

It is also possible to use the "render" keyword in a controller to move from one page to another once the controller is executed. For example, if the user tries to visit a page whose controller renders another page, then the user will ultimately end up on the page rendered by the controller.