**PHP (Hypertext Pre-processor) and MySqlAdmin compared to Ruby on Rails**

PHP compared to Ruby

Ruby is an object orientated programming language like other OOP’s it can do create object, classes, functions and methods as well as run code like mathematical operators, logical operators, arrays, if statements, loops, reading from files and more. On its own it is a general purpose programing language that is heavily focused on objects.

PHP is a server side programing language. It purpose is to run on the server, when it is executed it creates the HTML code for the result and sends it to the user, unlike HTML the user cannot see what code generated the result when they inspect the webpage. PHP can do more than this however as it can interact with a database(s) and run the way a general programming language can like ruby (or more closely to JavaScript) as well as keep the data stored in a session to be used again on a different page on the website and more.

So for developing code for a functional website that will collect data and output the HTML to the client PHP clearly wins that is what it is designed to do and Ruby does not seem able to do many of these feature required to make this work. However Ruby can work with Rails a Framework for web-application development.

Ruby on Rails Framework compared to PHP with other tools like MySqlAdmin

Rails is a Framework that handles a lot of the functionality of a website. This is great because it allows the developer to achieve most if not all the best practices of the functionality simple using the framework, this framework has the directories to handle the website.

Some of these directories are:

* stylesheets: contains CSS files for styling the website
* databases: contains SQL files to be run in a database
* views: contain the HTML files that give the webpage its layout or forms
* models: models are where most of the code logic based code will go
* controllers: controllers are used for navigating and deciding what views will be displayed

Creating a website without rails or fancy tools you can define your own directories and instruct the code where to look for other resources you want it to use. So for example you can tell it to look in another directory for the resource:

“<link rel="stylesheet" href="/html/styles.css">”

Or have it in the same directory and have it link directly:

“<link rel="stylesheet" href="styles.css">”

Rails allows you to use Ruby which allows you to treat the resource like an object. This means that it will be able to understand what you are referring to without telling it exactly where to look.

Ruby on Rails creates these resources for you as well by using command line commands like:

“ rails generate scaffold NAME [field[:type][:index] field[:type][:index]] [options]”  
“rails generate scaffold product name 'price:decimal{7,2}'”

Good IDE’s vs rails

. It uses controllers to determine what happens on that page. It decides what views should be called, this is useful for navigating the website.

Compared to PHP which I have used in previous years, navigation would work by links on the page, while this is still the case in ruby on rails every page loaded goes through the controller which can white or blacklist certain events like loading a page depending on the characteristic of an object like ‘current user’. A signed in user would still be presented with a sign in page if they navigated to that page in PHP, where as in ruby the controller would recognize the person has access and load the homepage instead and more importantly the other way around as well preventing access to pages without permission. There is ways to do this in PHP as well (redirects based on conditions) but require the developer program them into every web page, with the controller managing this in the Ruby on Rails framework it is easier to meet best practices.

Conclusion and Decision

<http://php.net/manual/en/intro-whatis.php>

<https://www.ruby-lang.org/en/about/>

<https://www.w3schools.com/html/html_css.asp>

<http://www.xyzpub.com/en/ruby-on-rails/3.2/scaffold_anlegen.html>