Lindsey Saari
Senior Project
Winter 2015
April 23, 2015

Final Essay

Introduction

When I first came to Northern Michigan University, I decided to major in Science and Math Education. After a few years in the program, I started to get second thoughts of a career in the Education field. I've always had a love for numbers and problem solving, so I started to brainstorm other majors that still involved Math. After speaking with friends and family, the idea of programming and Computer science was brought up and seemed appealing to me. Although during my high school years, I had never taken any sort of programming course (and none were offered), I felt that I could take on the challenge. I visited with Professor Andy Poe during the summer and based off of my math placement scores, he encouraged me to enroll in CS 120 to see if it would spark my interests.

Upon taking CS 120, I realized that I really enjoyed the challenge of programming and the problem solving skills that it required. After taking Computer Science courses for a few years, I was offered a position as a laptop technician at Micro Repair where I worked until I was offered a programming internship. The internship allowed me to work remotely and I was paid to learn Ruby On Rails. I picked up very quickly on the Rails approach to web design and learned how the Ruby on Rails framework has made

programming so elegant. The real life experience that I gained such as client interaction, working with co-workers, and the stages of project design are something that will definitely prepare me for my post college career in the workforce.

I feel that I have come a long way in the computer science program. Through all the struggles and learning hardships, I feel that each bit of struggle was a positive learning experience. I also have started to recognize my likes and dislikes as well as areas of interest as a programmer.

Project Overview

For my Senior Project, I decided to present something that I worked on during my internship. From speaking with various professors and students, I feel that not many people know enough about Ruby on Rails and people almost seem afraid when you mention the words. I think this may be do to their lack of knowledge and understanding.

With Ruby on Rails becoming one of the most popular web frameworks, I feel that it's important to be familiar with it. While learning Ruby on Rails, it was actually fun to compare and contrast the Rails approach with other languages such as Smalltalk and PHP. Although Rails is associated with a steep learning curve, I feel that it's not impossible to understand.

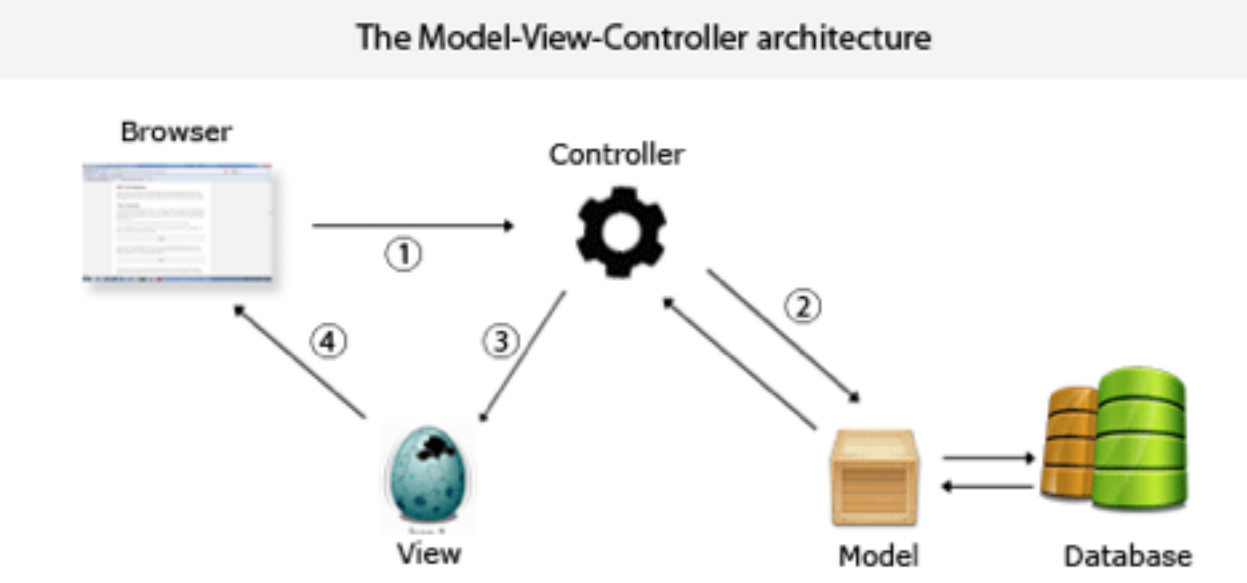
The project that I am sharing allows for construction companies to organize all the aspects belonging to a construction project. With user logins, users (with restricted roles) are able to log into the website and manage their projects. Once a user navigates to a project, the general setup section includes aspects such as budget and equipment

that belong to specific projects. The user is then able to edit each section accordingly. The project information is held in a sql database.

With the user in mind, it's important to see the project from the customers point of view at all times during development. This may have been the hardest part for me at times since I see things naturally from a developer point of view rather than how a user would see it. Testing was often difficult as well because as a developer, ones sees the "happy path" versus all the other ways that a user can interact with the application. Another difficult part was understanding all of the terminology and abbreviations that are associated with construction work.

Explain the Rails MVC

To explain the Ruby on Rails MVC briefly, we will start with the model. The model contains all business logic for the application and in this case is linked to a database. The view renders the UI and presents the data to the user. The controller receives events from the outside world (the view), interacts with the model and displays the view to the user. Below is a diagram in which the flow of the MVC is shown in a numbered sequence.



What I learned

One of the biggest things that I learned throughout the development process is that it's very easy to make something more difficult than it has to be. Ruby on Rails has libraries that you can include called "gems" which can be imported to the project. If there is something that you wish to do and you think that it has probably been done before, then there is probably an exiting gem that can help you to accomplish your task. Some of the bigger gems that were used for user logins were devise, cancan and rolify to help with user abilities, roles and authentication. Simple form is another gem that is very useful in relation to form submission.

Throughout the process i've become a better programmer. Often I would find myself mindlessly typing and then later go back to see that there were many lines of code that I didn't need, or could easily be condensed. For example, instead of using an if else statement, it's often more efficient to use the ternary operator. Rails helpers such as "exists?" and "blank?" made coding more efficient as well. I've learned how important it is to be able to critique and refactor your own code for efficiency purposes.

Rails follows the rule "Convention over Configuration." I learned this too to be very handy when it came to development. With the mapping between models and database tables, there is much to be said about the underlying naming conventions. This was very helpful once it was fully understood. I was also very happy that I had some previous knowledge of sql queries, which were used everyday.

jQuery was essential for traversing various elements in the DOM, and at first I was a little rusty, having not used it for some time. I went through a few tutorials and quickly relearned it. Now, jQuery is something that I fully understand and use everyday as well.

One thing that I learned about myself is that in order to improve as a developer, you need to set aside time to plan instead of rushing to write code immediately. I realized that most of the time, mindlessly writing code is inefficient and you usually end up wasting time. I've gotten a lot better at coming up with a plan first before I start coding. I also know that if I ever need help, it doesn't hurt to have a second pair of ears listening to your train of thought. There could always be something that you didn't initially think of that someone else did.

This was also my first time working directly with clients. I learned how to take constructive criticism and how important it is to understand what your client is really asking for. Often clients have no technical background, so it's easy to get frustrated with their lack of understanding. It's very important to have much respect on all levels and to make the client feel that what they are saying is important. If the client has your trust and you're delivering a good product, then the process stages should have better transitions.

Communication is also a key factor when it comes to client interaction. I remember one day when I was sending a client a report of my daily accomplishments and I got a very positive reply. They thanked me for being so thorough with my descriptions and my detailed questions. I was also applauded by my co-workers.

What would I do differently

If I could go back and do things differently, I think that I would slow down. Often when something wouldn't work, I would easily get frustrated and not get to the real root of the problem. I did though learn some great debugging approaches in the rails console

and in the browser. If there was a jQuery issue, for example, I could often paste my code into the browser console and check to see what was going wrong, or if there was some sort of rails issue, I could open up the rails console and test there. I also feel that I shouldn't have been so afraid to ask for help at times. I often thought that asking for help was showing that you were giving up, but in reality it's not. Asking for help is all part of the learning experience and it makes you a better programmer.

Organization

While using a sql database, the correspondence from the model to the database was quite simple. With the rails naming conventions, the project was very organized. I don't think that a rails project could ever be unorganized; I would say that it's almost impossible because the framework doesn't allow for sloppiness. For example, a controller action often corresponds to a view. If there was a controller called 'equipments' and there was an action called index, the index action corresponds to the equipments index view. The project is organized by models, views and controllers, so the naming conventions make it organized and easy to navigate. The routes also correspond directly to the specific actions. The most confusing part of the organization was the model relationships. Often the client could portray what they wanted, but it's the developers job to decide what relationships actually exist. The overall goal was to make the application easy for the user, and I feel like that goal was met.

What technologies/tools did you use, and why them?

The project didn't include any complex data structures or external technologies, but many other tools were used inside of the Ruby on Rails framework. Some of the tools that were used were rails gems, jQuery, bootstrap x-editable and jQuery data tables. I would also say that the Ruby on Rails framework was the overall technology used because it kept the project organized. The integrated RESTful routes were very helpful as well. Overall, The most important concept to understand was the MVC and the Rails Framework. Once that was understood at a high level, the overall flow of the project came easily.

What was the hardest part of the project?/struggles

One of the hardest parts of the project was working with the customer. As stated above, often the customer didn't have a technical background and the terminology used to describe specific details was a barrier at times. Having a customer that isn't exceptionally technical is difficult because they don't always understand the project from an internal standpoint. As a developer it's difficult to describe what object design is to someone who isn't technical.

One of the most difficult parts for example, was when the user suggested that they wanted to be able to update the location of equipment inline without having to navigate to a form. I researched many options and came up with bootstrap x-editable for inline editing. This allowed for a user to select an equipment location and update it by selecting from some drop down of options. I ran into a problem when I realized that I had to manually submit the parameters to the database. After developing a method, I

thought that I was in the clear, but then ran into another issue. When I was trying to perform a “PUT” request, when submitted through ajax, the request method wasn’t being preserved and got switched to a “GET” request, by default. After much research, I realized that I had to preserve the request method upon redirect by passing a “303” status as a parameter, but it took much effort to get the feature to work.

Another difficult part of the project was the relations between objects. For example, if a project has many equipments, you could code something such as, “project.equipments.all” to obtain all of the pieces of equipment, but if you forgot to make equipments plural, then the program would crash. Often the smallest errors are the most difficult to see. Ruby on rails is all about naming conventions and often something so simple could be your biggest fault. Even though the naming conventions gave me trouble sometimes, I don’t have much negativity on how the MVC flows. It makes navigation and project organization entirely simple.

Another part that was difficult was the jQuery data tables. Data tables is a library that can be included in the application javascript file. The reason for data tables is for a user to have the ability to search, sort, or page through sets of data in a table. I ran into a problem when I was trying to incorporate data tables because it required an updated version of jQuery. What I ended up doing was updating the jQuery version for the entire application. By updating the jQuery version, other pieces of the application that were depending on the earlier version started to break.

For example, I remember certain aspects getting “buggy”, such as the icons on all the date picker calendars disappeared. As more bugs and glitches started to appear during testing, I was informed of how it was important to check with others before

updating such big libraries such as jQuery versions. I then had to look into all the issues associated with the updated version. Now I am more cautious when including and updating libraries, as it may cause serious application bugs.

Was the project about as hard as you predicted? If not, where was the error?

I think that the project was a good level of difficulty. There were times where I would get frustrated, but my greatest resource was the help of others who would get me to draw out my thought process. Even though we live in a paperless world today, I often found that my best development came from using a paper and pencil to draw out my design. Along the way, i've realized how far i've come from when I first started to learn Rails. It was also nice having previously learned PHP, and learning how Rails compares and contracts to PHP and other languages. I hope that at some point more students will be able to get some experience with the framework as Ruby on Rails becomes more popular.

Citations

Tamada, Srinivas. The Model-View-Controller Architecture. Digital image.

Getting Started with Ruby on Rails. 8 Feb. 2011. Web.