Joseph Lee
Melynda Lindhorst
CS 4540

Ruby/Rails Project: Dungeons and Dragons Loot Generator

Abstract

In Dungeons and Dragons, when the players successfully kill an enemy or accomplish a goal or quest, they receive loot in addition to experience and money. Loot generation is a task that falls on the dungeon master. It requires a few calculations followed by cross-referencing what the calculated numbers actually reward the players with in their current game environment. This is a tedious task that is easily parsed and solved by a computer. The Dungeon and Dragons Loot Generator, part of a future larger Dungeon Master Toolset, aims to do just that.

Tutorial

The home page of our website displays a navigation bar and title, which are persistent across the entire website. You'll also see a textbox to put in your party's average level. After doing that, it will ask you to pick a die and roll it. This will bring up tables with entries that correspond to your roll and the level you've chosen. All of the entries listed are acceptable loot for dungeon masters to reward their players with. You will also see accordions listing all of the possible loot rolls you could have had.

The navigation bar links you to pages that allow you to edit the entries for each individual loot table. You are allowed to view, change, create, and delete entries to better fit your Dungeons and Dragons experience. Once you're done editing the tables, all future die rolls will take the new data into account for loot generation.

Joseph Lee
Melynda Lindhorst
CS 4540

Software Overview

Dungeons and Dragons Loot Generator was built in conjunction with Rails' strict adherence to MVC. We maintained both the MVC file structure and communication throughout development. POST was used to transmit data from the controller to the views. Similar to creating PHP websites, we used markers to insert values or function calls into the html documents. We also used the shorthand model features in Rails to allow for quick and easy setup for our create, edit, and delete pages. We also used the built-in query function calls to query our SQLLite database without needing to write any SQL code. Bootstrap was used along with custom CSS, JavaScript, and JQuery to create the front-end views for the website. The majority of the testing for Dungeons and Dragons Loot Generator was done manually.

Individual Contributions

- Joseph Lee: For my contribution, I set up the ruby on rails project and created the base model code. I also set up the index pages and laid the data out. I created the randomization and loot rolling helper functions. Then I wired the pages together with links to allow access between the pages. I focused mainly on the Ruby portion of the project.
- Melynda Lindhorst: For my contribution, I spent a lot of time with the front-end of our website. I focused on mixing custom CSS with Bootstrap to obtain the sharp and clean look and feel on the pages. I also used JQuery to display our data in a dynamic way; we didn't want to display multiple large tables all at once. I also assisted in connecting the data from our Ruby back-end to the front-end display.

Joseph Lee
Melynda Lindhorst
CS 4540

Summary

Like PHP, Ruby requires a different way of thinking. It's difficult to overcome the disparity between object-oriented languages and scripting languages, especially when the opportunities to utilize scripting languages are so sparse in undergraduate academia and we have little previous experience to work off of. Nonetheless, this was a rewarding project to work on and the Dungeons and Dragons Loot Generator is something that both of us will probably improve upon and add to in the future. Our Loot Generator falls into the Good to Superior range because we began knowing nothing about Ruby, but we spent a lot of time and effort to learn its operations and quirks and the resulting product is robust, fun, and useful.