Simply Written Functionality

by David Turner

Module Code: AAD 716
Module Title: Masters Project

Introduction

Simply Written is a complex service, built up of many different components that all come together to provide a service built from the ground up to make it easy for authors to create, and publish, digital books that are of a consistent high quality, both in terms of content and in terms of the book itself.

In this document I want to outline some of the functionality that allows this to happen, and delve into how it works.

Overview

Simply Written is made up of four different sections, all of which contribute to the site working in a manner that can create high quality digital products. Those four section are:

- Content Authoring Site
- Image Storing
- ePub Generation
- Mobi Genration

Each of these focuses on a single aspect of the product, allowing them to be handled in a manner that is best suited to the problems each has to deal with.

Content Authoring Site

This is the only area of the site that users interact with directly, the area of the site that is used for creating and editing digital books. As such it handles the vast majority of the functionality of the Get Invited service, and has a wealth of functionality built into it. Some of these areas are of significant importance and it is these areas I wish to expand upon briefly.

The functionality that I wish to cover in this document that relates to Content Authoring consists of:

- Creating a New Book
- Creating a New Chapter

There is much more functionality to the Simply Written service, but these are the two key parts of the service upon which authors depend.

Creating and Editing Books

Creating and editing is the first time that a user will have to do upon registering for an account. The act of publishing a new book can require a great deal of information, but when you are working on a book, much of this information has yet to be decided upon.

When creating, or editing, a book Simply Written only requires one piece of infromation, a Title. Everything else is optional, though the service provides some safe defaults for certain details.

By default only the Book Title and Book Description are in view for users, a selection of additional information can be added to books by viewing the optional fields. These hidden fields serve to handle many of the legal aspects of books, along with some additional information that may be of relevance.

Once a book has been created it becomes possible to upload a Book Cover. This cannot be done before a book has been created as important information, from a technical standpoint, is not available. I will cover this when we get to the section on image storing.

Creating and Editing Chapters

As with the process of creating and editing books, the process of creating and editing chapters is very similar, and is even more simplistic. The only information needed in a chapter is the Chapter Title and the Chapter Content.

Content is saved in a format called Markdown. This decision has been thought through and the format has proven to be the best format for storing content for a couple of reasons:

- It is flexible
- It helps remove messy code

Markdown is flexible as it allows for output into different formats, which afford Simply written users the maximum amount of flexibility for output formats. This means that, from one central source of content, books in multiple format can be generated. This is the basis that allows output in both ePub and mobi formats, and also provides the flexibility to consider PDF generation in the future.

It also helps to remove messy or otherwise damaging code. This is a serious problem that sadly affects a large majority of digital books currently available. Poor code results in books becoming bloated. This bloat results in a poor reading experience for those who purchase the books. By storing content in a format that is code agnostic it becomes quite easy to ensure that content across formats is tailored specifically to those formats.

Image Storing

Image Storing is something that was not in the initial release of Simply Written. Image files are reasonably easy to handle, and are quite easy to store. Storing them in a manner that allows them to be easily included in digital books, however, proved to be quite difficult.

This was a puzzle in two different parts. The first part was storing the images in a manner that was accessible to the digital book generators used by Simply Written, and the second was storing them in a manner that allowed the author to see the images they were using in the book as they wrote it.

I had already partially solved this problem in the initial release of Simply Written, as book covers worked in a manner that could be viewed in the service. Unfortunately this functionality was very simplistic, as the book cover does not need to be embedded into the book at any point other than when the finished file is generated.

I was able to abstract this information into a system that allows for images to be uploaded and saved in a manner that can be accessed with relative ease. There are a couple of pieces of information that were needed for this to work:

- Author ID
- Book ID

These two pieces of information allow the system to store books in a manner that ensures each image is saved in a location that is relevant to only a single book. As an added bonus this information is available throughout the service, making accessing these images quite easy.

The only issues that remained with images was to prevent images from being overwritten. This is handled as images are uploaded, as each image in renamed depending upon the time at which it is uploaded. This ensures that no image will ever be accidentally replaced.

With this system in place it became easy to upload, and to access, images. The system for accessing images became:

/author/book/file

An example of this would be:

/1/2/0123456789.jpg

This means that a book with an author with an Author ID of 1, and a Book ID of 2, was trying to access a file named 0123456789.jpg.

ePub Generation

ePub genration served as the big issue of generating digital books, as it is made up of multiple different components coming together perfectly. An issue with any single part of the process causes a book to simply not work.

It took quite some time to understand the pitfalls of the format, and even more time to figure out how to generate valid ePub files on demand. The biggest issues were caused by a mere two files:

- content.opf
- toc.ncx

These files are fundamental to any ePub file, as the former tells eReaders what content is there, and the latter deals with a book's Table of Contents. Both of these files are based on XML, a highly flexible format. The format is quite strict in handling errors however so any errors in generating these files results in errors in the generated ePub file.

Working through these issues resulted in a firm understanding of the technical aspects of genrating ePub files and an ability to highly streamline the process of generating the files needed to create a digital book. This paid dividends when it came to generating mobi files

Mobi Generation

Mobi files proved to be an interesting challenge, as the process of generating them is shrouded in mystery. The format is owned by Amazon and used exclusively for their Kindle devices, and they do very little to explain how to create a .mobi file. The do, however, provide a tool you can use to convert other content into .mobi files.

This allowed me to convert the content I would otherwise use in an ePub file into a .mobi publication, though there were a few bumps along the way. Whilst Amazon's tool did convert the files my ePub Generator created, it had two issues:

- Duplicated Cover
- No Table of Contents

Kindle books, it turned out, automatically insert the cover into the book, which allowed me to cut some content from the ePub content generated to make .mobi files, something I was only able to work out because of my time spent working on ePub generation.

The Table of Contents needed to be manually generated for the Kindle. Unlike ePub, the Mobi format doesn't make use of the toc.ncx file at all. This allowed me to, again, cut down on some generation, and resulted in a slightly quicker generation of .mobi files.