Flyyta - A static site generator

K.C College Of Engineering and Management Studies and Research
Abhishek Mogaveera[1], Devika Olkar[1] and
Assistant Prof. Nilima Patil[1]

Abstract. Static websites are still top-rated in the business environment. They offer ease of development, present low security risks, and can generate value rather quickly due to informational content being quite significant. It helps with building a reputation amongst the target audience and generating new leads. With the rising need for content marketing, dynamic websites became popular as they allow one to present actual information. All of this comes at the cost of impacting page loading speed that is also critical. Static site generators (SSGs) aim to solve this issue. SSG is a tool to build a static website from source files. It adds more flexibility to static websites allowing them to react to dynamic content changes, regenerate, and publish again. The proposed system is an implementation of static web-pages generator. Using this method allows users to save time to make the site so that they have more energy to take into account customer's needs and interests.

Keywords: Static site generator, JavaScript, Markdown

1 Introduction

As websites grew to deliver more and more content, the web development industry found new ways to make the process of maintaining and updating sites more efficient. Web servers would perform that task on demand whenever a request for a resource was received. They'd faithfully combine templates and content, apply loops and logic, and return a page view whenever one was requested. With the rising need for content marketing, dynamic websites became popular as they allow one to present actual information. All of this comes at the cost of impacting page loading speed that is also critical.

Static site generators (SSG) do much the same thing. They apply data and content to templates, and generate a view of a page which can be served to the visitors of a site. The greatest difference between a static site generator and a traditional web application stack, is that instead of waiting until a page is requested and then generating its view on demand each time, a static site generator does this in advance so that the view is ready to serve ahead of time. And it does so for every possible view of a site at build time. The generated pages are then deployed to a web server. When the server receives a request, it responds with rendered HTML. Static pages eliminate the latency that databases introduce.

Based on the above problems we have proposed a static site generator "Flyyta". The key features of Flyyta include a Command Line Interface that will help the user seamlessly create, build and deploy the project. Flyyta also comes in with some beautiful templates that the user can build upon. This helps the users right from

mediocre to expert level in web development to create and host their own website as quick as possible.

2 Literature Review

The need of having websites is increasing day by day, for this, we have gathered information from various sources. “Static Web-Pages Generator Using JavaScript” whose authors are Wenrong Jiang, Jihong Yan2.Web page generators mainly use JavaScript as a scripting language. JavaScript is widely used on the client-side. The web scripting language has developed by the use of JavaScript to develop interactive Web pages. The emergence of JavaScript makes web pages and users realize a real-time, dynamic, interactive relationship, in order to make web pages more interactive, JavaScript scripting language is essential, and it is very important. It explains an approach to generate a web page using JavaScript. “Blogging platform utilizing Kentico Cloud and Jekyll” static site generator whose authors are Radoslav Karlík (2016). A modern CMS is more than just a content repository. It excels at supporting more content-consuming channels where the static website generated by the SSG could be only one of them. It also provides tools to manage the whole content life cycle like reviews and collaboration. Currently, not a lot of SSGs support integration with a headless CMS system. This aims to integrate Kentico Cloud as a headless CMS into the Jekyll SSG framework. The goal is to explore the Ruby ecosystem and discover potential problems of using Jekyll together with a headless CMS. Jekyll framework was chosen because of insufficient solutions that would integrate a headless CMS, despite being one of the most popular. Its present advantage is that it does not need a CMS, and all content can be written in Markdown and stored in a GitHub repository. “Design, Use and Business Model.MOOCs (Massive Online Open Courses)” (2017) are shaking up the traditional forms of primary and continuing education and training. It explains that the common ground of the community is the love of online learning. Web development and content management often help in building the innovators and allowing the community to stay together and build the user-friendly GUIs. “edX E-Learning Course Development”, written by Matthew A. Gilbert, MBA. EDX E-Learning Course Development, aim on designing, developing, and deploying MOOC courses on the edX platform. It helped in some way in designing this project following some common good practices as mentioned by the author. “Moodle 3 E-Learning Course Development” written by S. S. Nash, E. Rice.Moodle is a learning platform or Course Management System (CMS) that is easy to install and use, but the real challenge is in developing a learning process that leverages its power and maps the learning objectives to content and assessments for an integrated and effective course. Moodle 3 E-Learning Course Development guides you through meeting that challenge in a practical way. How to create and integrate third-party plugins and widgets in your Moodle app, implement site permissions and user accounts, and ensure the security of content and test papers was conveyed here. “Building Web Apps with WordPress” written by B. Messenlehner, J. Coleman.WordPress is a full-blown framework capable of many things. Additionally, WordPress is built on PHP, JavaScript, and MySQL technology, so anything you can

build in PHP/My SQL can be bloated into your WordPress application easily enough. It proposes two approaches for easy creation and maintenance of SPOC. “Enterprise Web Development” written by N. Abbott, R. Jones, M. Glaman, C. Chumley. Build complete, complex websites with no prior knowledge of web development entirely using an intuitive user interface It ensures our sites are modern, responsive, and mobile-friendly through utilizing various features. Quickly master theme administration, custom block layouts, views, and template structure. Helped to build modern, scalable websites. “Foundation of Joomla” written by B. Harwani.Foundations of Joomla is a step-by-step practical guide that explains building websites and blogs using Joomla - a very popular and powerful content management system (CMS). It explains the steps of installing Joomla, configuring your database, creating a blog and a website, followed by instructions on creating new posts and adding content to your site. One can build responsive, powerful, and fully featured websites quickly. It explains the foundations of Joomla which is a CMS. “Beginning Django CMS” written by Nigel George. Beginning Django CMS shows how to simply and easily write a dynamic website with a full content management system in the backend. It is written for Internet developers who are sick and tired of dealing with complicated, bloated website frameworks that are a pain to build and a nightmare to maintain. Django CMS is an Open Source website building framework that is experiencing exponential growth because it is built on the simple, secure, and scalable architecture of Django. We get well-informed knowledge of the Django framework, to build a functional website and content management system that you can deploy for your own website or for your customers.

3 Architecture

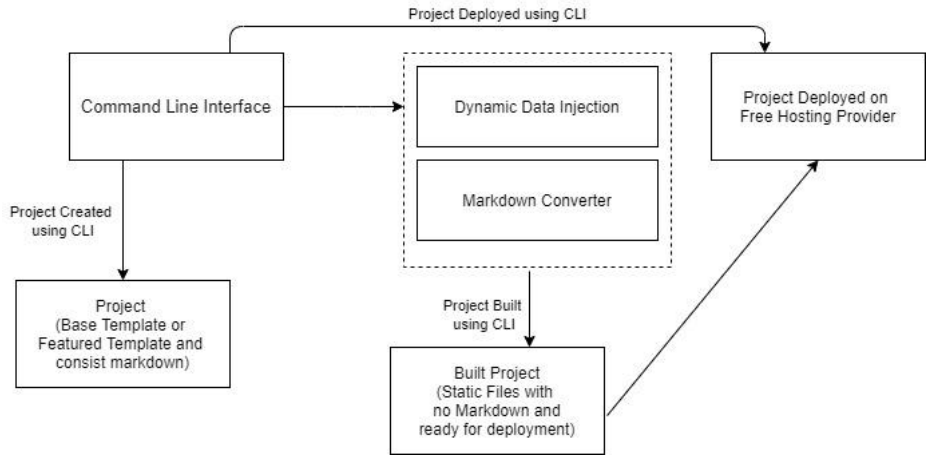

Fig. 1. This shows figure of the architecture of the system.

4 Description of the Architecture

The system proposed consist of components : a Command Line Interface and the project built by it which consist of smaller modules in itself. The Command Line Interface is a program that takes in commands, which it passes on to the computer's operating system to run. This command line tool helps the user create ,build and deploy project. The other component is the entire project that is generated by this tool which will hold the structure and the files. The entire project architecture consist of small modules like the Command Line Interface, the markdown converter and the file convertor.

4.1 Command Line Interface (CLI)

A command line interface (or CLI) is a text-based interface used for entering commands. This interface is ready to accept a command from the user to create, build or deploy a project. This Command Line Interface will use a node package which has to be globally installed . With the help of this package the user will be able to run commands from the command line . The Command Line will give the user commands to create a project with no or with some template to make development faster, build commands to build the project and deploy option to deploy the project to a free hosting provider.

4.2 Markdown converter

Markdown is a lightweight markup language for creating formatted text using a plain-text editor. Markdown is widely used in blogging, instant messaging, online forums, collaborative software, documentation pages, and readme files. Flyyta accepts Markdown files as a source and then that is converted and the data is injected into HTML. We are using a node package called marked which is a low-level compiler for parsing markdown without caching or blocking for long periods of time. When the project is built the markdown files are converted and thus the output is simple HTML. This module is also using another node package called front matter for extracting and parsing front matter from strings.

4.3 Dynamic Data

The data that is injected into the HTML is dynamic and is entered by the user. But as HTML does not support variables, we have created a system to inject dyamic data into HTML. Once the project is built, the data is injected and the output is HTML file with the dynamic data . We are using regular expressions to target the variables in HTML enclosed with {} and then injecting the variable value assigned by the user into it.

4.3 File Conversion

The files are a mixture of HTML,CSS and JavaSript if any and markdown files used majorly for blogs. The process begins with the the dynamic data injection as mentioned above and then the markdown is converted and thus the project is built and the output is static files with no markdown.

5 Conclusion

The goal of this paper was to propose a system to implement and build a Static Site Generator and also outlined the basics of web development and technologies, followed by explaining the various approaches for a generation of web pages and save time not creating bloated websites. Additionally, various API's can be created so that this Static Site Generator can be integrated with other platforms to provide a much better and easier experience to the user.

References

[1] Wenrong Jiang, Jihong Yan2.Implementation of Static Web-Pages Generator Using JavaScript, 2011

[2] Radoslav Karlík, Blogging platform utilizing Kentico Cloud and Jekyll static site generator, 2019

[3] J.-C. Pomerol, Y. Epelboin, C. Thoury MOOCs. Design, Use and Business Model. London: ISTE Ltd., JohnWiley & Sons, 2015

.[4] F. Fox. From MOOCs to SPOCs. Communications of ACM, vol.56, #12, p.38-40, 2013.

[5] J.M. A. Gilbert. edX E-Learning Course Development. Birmingham: PACT Publishing, 2015.

[6] S. S. Nash, E. Rice. Moodle 3 E-Learning Course Development, Fourth Edition. Birmingham. GB: Pack Publishing, 2018

[7] B. Messenlehner, J. Coleman. Building Web Apps with WordPress. Sebastopol, CA, USA, O'Reilly Media, 2020.

[8] N. Abbott, R. Jones, M. Glaman, C. Chumley. Drupal 8: Enterprise Web Development. Birmingham: PACT Publishing, 2015.

09] B. Harwani. Foundations of Joomla. Apress, N.-Y., USA, 2016.

[10] Nigel George. Beginning Django CMS. Apress, N.-Y., USA, 2015