UNITECH SELECTED PAPERS

International Scientific Conference
UNITECH`2024

PERFORMANCE COMPARISON OF WORDPRESS POSTS AND MYSQL DATABASE

Dijana Stojić[1*], Dejan Vujičić[1], Đorđe Damnjanović[1], Dušan Marković[2]

[1]Faculty of Technical Sciences Čačak, University of Kragujevac, Serbia;
[2]Faculty of Agronomy Čačak, University of Kragujevac, Serbia;

**Corresponding author: dijana.stojic@ftn.kg.ac.rs*

Abstract

Nowadays, WordPress is increasingly used for creating modern websites. An advantage of using WordPress is its built-in CMS for adding new posts and pages. More complex data collections, such as a German language collocation dictionary, require the existence of a database. The use of a database for loading the dictionary could be avoided by using custom posts in WordPress. This paper tested the loading of data from the dictionary using the Data Tables HTML JavaScript library, which allows for good content overview and enables searching across any field in the table, which is very useful for dictionaries. The loading was performed in two ways: the first method was through posts within the WordPress site, and the second method was directly from the database. Parameters of loading were measured with different numbers of entries, and comparisons were made.

Keywords: Data Tables HTML Javascript, database, performance testing, posts, WordPress.

INTRODUCTION

Creating simple WordPress [1] sites does not require a high level of programming knowledge. More and more developers are using WordPress due to its ease of content creation and maintenance. The built-in CMS (Content Management System) simplifies the process for site editors to add new pages or posts, reducing the workload for developers during site maintenance, which in turn minimizes maintenance costs. This paper tested the loading of a large amount of data onto a WordPress site in two ways: using posts and a MySQL database. Comparisons were made of different performance metrics when loading the created pages.

In the second chapter, the method for creating test pages is presented, while the third chapter provides the testing results and discussion. The final chapter presents the conclusions and needs for further improvements.

EXPOSITION

The need to test different methods of data loading arose from the creation of a German language collocation dictionary within the DeSKoll project [2]. For creating a tabular display, the Data Tables HTML JavaScript library [3] can be used, which efficiently presents data obtained in various ways. For testing purposes, a dictionary of English words with a simple structure [4] was used. It consists of four columns: Word, Count, POS, and Definition. For our testing, the specific meanings of the columns were not important. Rows of 5,000, 10,000, and 20,000 entries were loaded in two ways. The maximum number of rows was set to 20,000, as the specific collocation dictionary we aim to create will not exceed that number. Identical pages were created on the WordPress site for both loading methods, as shown in Fig. 1.

doi: www.doi.org/10.70456/.....

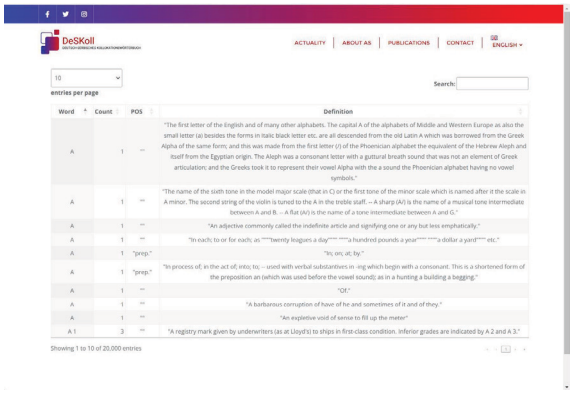

Fig. 1. *The appearance of the loaded web page*

The first method of loading was executed using posts, with one post created for each row. An example of a loaded post is shown in Fig. 2. [2]. This method is suitable for potential later modifications of certain rows by the site editor. The second method of loading data is directly from the database. For both methods of data storage, loading from a CSV file was used.

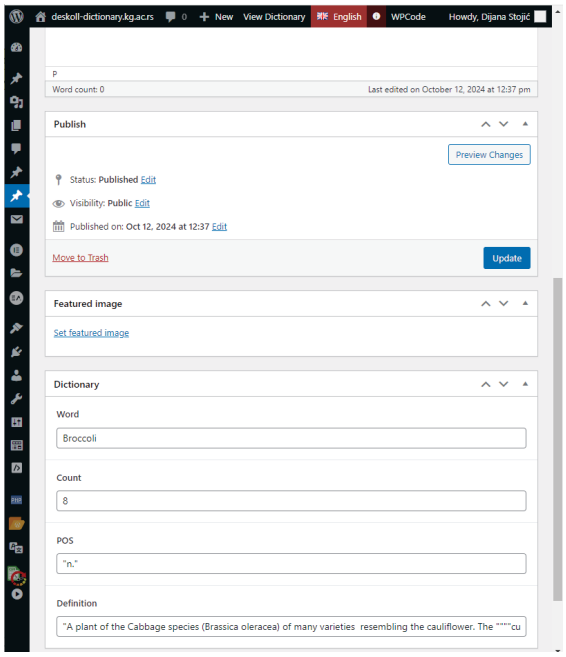

Fig. 2. *The appearance of the loaded post*

The method of loading data from the database into the Data Table is shown in Fig. 3.

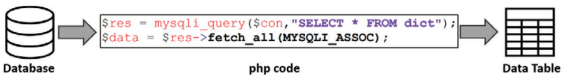

Fig. 3. *Loading data from the database into the Data Table*

The method of loading data from WordPress posts into the Data Table is shown in Fig. 4.

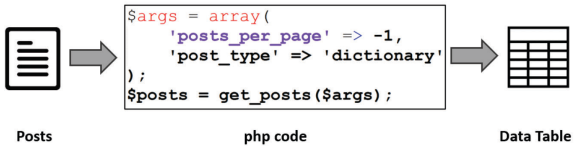

Fig. 4. *Loading data from WordPress posts into the Data Table*

We wanted to compare whether loading via the database is significantly better than loading via posts, as editors cannot access the database itself. As shown in Fig. 2, by using built-in WordPress posts, editors can easily make changes visually without needing to understand the underlying data structure. This cannot be said for the MySQL database, as some programming knowledge is still required.

RESULTS AND DISCUSSION

Performance measurements of the pages were conducted using a Python script, utilizing Selenium [5] and Chrome Web Driver [6]. A part of the script is shown below:

```
# Page load time
start_time = time.time()
driver.get(url)
end_time = time.time()

# Total page load time (when
the page is fully loaded)
load_time = end_time -
start_time

# Site performance
performance_data =
driver.execute_script("return
window.performance.timing")
dom_content_loaded =
(performance_data['domContent
LoadedEventEnd'] -
performance_data['navigationS
tart']) / 1000
total_load_time =
(performance_data['loadEventE
```

```
nd'] -
performance_data['navigationS
tart']) / 1000

# Time to Interactive
measurement
tti =
driver.execute_script("""
   let entries =
window.performance.getEntries
ByType('navigation');
     if (entries &&
entries.length > 0) {
        let timing =
entries[0];
        return
(timing.domInteractive -
timing.startTime) / 1000;
             }
     return null;
   """)

return {
     'Total Load Time (s)':
total_load_time,
     'DOM Content Loaded Time
(s)': dom_content_loaded,
     'Page Fully Loaded Time
(s)': load_time,
     'Time to Interactive
(s)': tti if tti else 'N/A'
}
```

The script measured several parameters significant for evaluating the loading performance of web pages for different numbers of loaded entries (5,000, 10,000, and 20,000) [7-13]:

- *Total Load Time (in seconds):* represents the total time required for the page to fully load in the browser. This includes the time needed to load all elements on the page, such as HTML, CSS, JavaScript, images, and other resources.
- *DOM Content Loaded Time (in seconds):* indicates the moment when the HTML document is fully loaded and parsed, and all DOM elements are available for manipulation via JavaScript.
- *Page Fully Loaded Time (in seconds):* indicates the time required for the web page to fully load, including all resources such as images, scripts, styles, and all other elements.
- *Time to Interactive (in seconds):* indicates the time required for the web page to become interactive, meaning when all essential resources are loaded and the DOM is ready for user interactions.

Fig. 5. shows the measurement results for the Total Load Time parameter for all three cases.

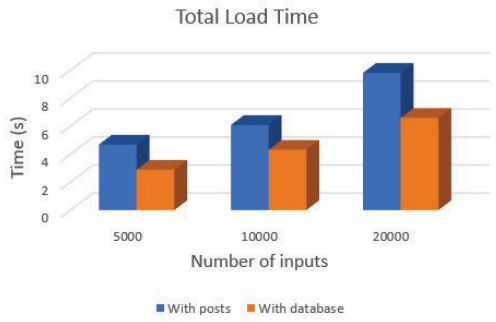

Fig. 5. *Measurement of the Total Load Time parameter*

Fig. 6. shows the measurement results for the DOM Content Loaded Time parameter for all three cases.

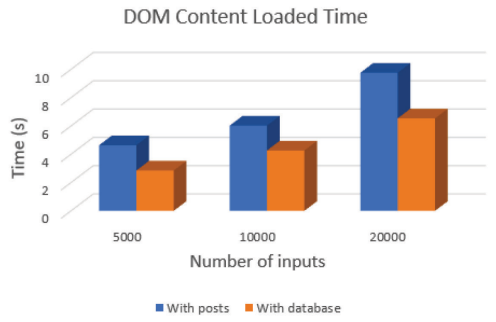

Fig. 6. *Measurement of the DOM Content Loaded Time parameter*

Fig. 7. shows the measurement results for the Page Fully Loaded Time parameter for all three cases.

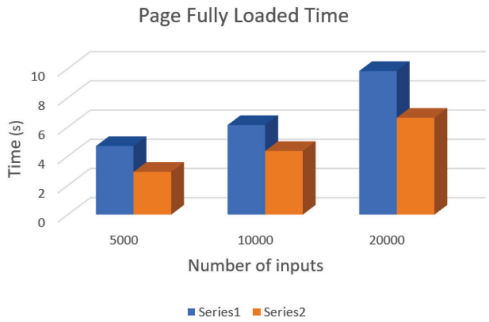

Fig. 7. *Measurement of the Page Fully Loaded Time parameter*

Fig. 8. shows the measurement results for the Time to Interactive parameter for all three cases.

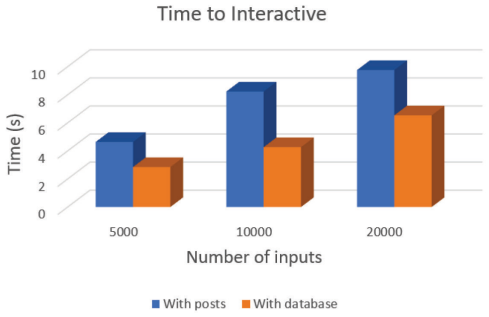

Fig. 8. *Measurement of the Time to Interactive parameter*

The results obtained showed that performance varies for different quantities of loaded data, and as the number of data entries increases, the differences become more pronounced. This can be partially explained by the fact that, although posts are stored in the built-in WordPress MySQL database, it takes longer to load them into the data tables. When entering data directly from the database into the table, data is accessed directly, whereas when entering data from posts, data is accessed indirectly by calling the post, which retrieves the data from the WP MySQL database.

CONCLUSION

This paper tested two methods of loading a large amount of data onto a WordPress site. The first method, loading via posts, proved to be good for later modifications by editors, but performance results showed that it is significantly less favorable in terms of site efficiency. The second method, loading via the database, is better in terms of performance. Four time metrics relevant to the loading performance of the web page were measured.

Since editors are not allowed access to the database, a page should be created within the site that would enable editors to modify any incorrect data without involving programmers. This page could provide a customized interface for data entry and editing, thereby increasing the efficiency of editors' work and reducing dependence on technical staff. Considering these aspects, it is clear that there is a need for a balance between performance and data accessibility for editors when loading large quantities of data onto websites.

Acknowledgments: This research was supported by the Science Fund of the Republic of Serbia, PROMIS, Grant no. 10916, German-Serbian Collocation Dictionary for German Language Learning and Teaching – DeSKoll. Also, by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia, as parts of the Grant No. 451-03-66/ 2024-03/200132 with University of Kragujevac – Faculty of Technical Sciences Čačak, and Grant 451-03-66/2024-03/ 200088 with University of Kragujevac – Faculty of Agriculture Čačak.

REFERENCE

[1] WordPress: https://wordpress.com/, accessed on October 16th 2024.
[2] DeSKoll Dictionary: https://deskoll-dictionary.kg.ac.rs/, accessed on October 16th 2024.
[3] Data Tables: https://datatables.net/, accessed on October 16th 2024.
[4] The Online Plain Text English Dictionary (OPTED): https://www.kaggle.com/datasets/dfydata/the-online-plain-text-english-dictionary-opted?resource=download, accessed on October 16th 2024.

[5] Selenium: https://www.selenium.dev/, accessed on October 16th 2024.
[6] Chrome Web Driver: https://developer.chrome.com/docs/chrome driver, accessed on October 16th 2024.
[7] Schmidt, Kristi E., Yili Liu, and Srivatsan Sridharan. "Webpage aesthetics, performance and usability: Design variables and their effects." Ergonomics 52.6, 2009, 631-643.
[8] Mohammed, Rahimoddin, et al. "Optimizing Web Performance: Front End Development Strategies for the Aviation Sector." International Journal of Reciprocal Symmetry and Theoretical Physics 4, 2017, 38-45.
[9] Chiew, Thiam Kian. Web page performance analysis. Diss. University of Glasgow, 2009.
[10] Ramakrishnan, Raghu, and Arvinder Kaur. "An empirical comparison of predictive models for web page performance." Information and Software Technology 123, 2020, 106307.
[11] Butkiewicz, Michael, Harsha V. Madhyastha, and Vyas Sekar. "Characterizing web page complexity and its impact." IEEE/Acm Transactions On Networking 22.3, 2013, 943-956.
[12] Zhou, Junzan, et al. "Predicting web page performance level based on web page characteristics." International Journal of Web Engineering and Technology 10.2, 2015, 152-169.
[13] Xilogianni, Christina, et al. "Speed matters: What to prioritize in optimization for faster websites." Analytics 1.2, 2022, 175-192.