UDC 004.051 DOI: https://doi.org/10.32782/1995-0519.2022.1.16

CORE PHP AND PHP FRAMEWORKS: COMPARATIVE ANALYSIS OF TWO APPROACHES FOR BACKEND DEVELOPMENT

Marian Slabinoha
PhD in Engineering,
Associate Professor at Computer Systems and Networks Department
Ivano-Frankivsk National Technical University of Oil and Gas, Karpatska str., 15, Ivano-Frankivsk, Ukraine, 76019, marian.slabinoha@nung.edu.ua;
ORCID: 0000-0002-7296-0356

The purpose of this paper is to compare two approaches of backend development using the PHP programming language – Core PHP and PHP frameworks. Both approaches are popular, but, as in every other language, using frameworks saves developer time and allows him not to implement routine connected with the trivial operations.

Still, when we are talking about performance and "weight" of the web-application, we can say that using frameworks often includes a lot of redundant and unnecessary code from packages that are used, and a lot of web-developers don't control that. As a result, the developed web-applications become heavier and consume more resources of server systems. Moreover, frameworks approach is often used for very simple applications that can be easily implemented using Core PHP and based on a few files of code. This creates unnecessarily large projects solving the small and simple tasks.

The aim of this paper is to highlight the advantages and disadvantages of PHP web development using Core PHP and PHP frameworks. The security and performance are investigated mostly, but some smaller things are taken into account as well. The recommendations on using both approaches are given to provide most effective combination of development experience and performance.

As the result of comparison analysis, the framework-like MVC application architecture is proposed to build tiny lightweight web applications for custom tasks (in our case, the vocabulary website). The structure of the application is provided and key development things are highlighted (using Core PHP).

The perspectives of modification for solving trivial problems like RESTful backend app or bearer token authentication are discussed.

Key words: sustainable web development, performance optimization, PHP frameworks, MVC architecture, Core PHP.

PHP has been one of the most popular backend developments for a long time. LAMP stack as the platform configuration remains the most popular way to host the web-sites. According to W3Techs [1], 77.4 % of websites whose server-side programming language is known, use PHP.

PHP started as the scripting language, but soon became the powerful tool for the "classic" programming approach. As a result, a lot of different frameworks that used PHP as a basic programming language were developed and released. Of course, this made developers work easier and less time consuming, providing all-in-one popular functionality of the application.

However, the performance of the PHP frameworks remains questionable, compared to Core PHP. Moreover, a lot of small backend web applications are written using PHP frameworks and include a lot of redundant code from different packages, so most functions from these packages remain unused.

The aim of this paper is to compare two approaches and give the recommendations regarding usage of each approach depending on the project characteristics.

So, when we are talking about the difference between using Core PHP and PHP frameworks, a couple of things should be considered. First of all (and this is the main purpose of the frameworks) is the convenience of the development process. PHP frameworks are known to be time-saving while developing large applications [2]. The reason of this frameworks approach advantage is including the packages that solve standard web development routine in your project without having to write it from scratch. This includes the authorization process, routing and using the database as an abstraction. Developers can simply use the pre-developed functions without writing them by themselves.

Second big advantage of the framework is team interaction during the development process. Core PHP requires the developer team to agree on a lot of small and big things – how to break the app structure into files, packages or object-oriented programming entities, etc. In the case of a framework you already have the skeleton of your project and stick to the design recommendations of that framework.

Easy maintenance and scalability is the next advantage of PHP frameworks [3]. The requests,

loading balance and many other things connected with the app performance are already developed and configured when using frameworks. This allows you to build relatively big and complex applications without worrying about how it will behave when the number of customers increases.

When talking about advantages of using Core PHP, the first one that comes to mind is controlling all your code. When developer builds the application from scratch and does it in the proper way, using KISS (Keep It Simple Stupid) and DRY (Don't Repeat Yourself) principles, he gets the application that is fully transparent and traceable. This allows the developer to rely on himself at major part of the code and not to worry about packages updates, version conflicts, etc.

But the key advantage of the Core PHP is performance. Author of the paper [4] gives the detailed analysis of CRUD (Create – Read – Update – Delete) operations performance for Core PHP and popular frameworks – Phalcon, CodeIgniter, Laravel and Symphony. According to the research, the fastest framework (Phalcon) application executed CRUD operations more than 2 times slower than Core PHP application, while the most popular PHP framework, Laravel, was up to 400 times slower than Core PHP.

Worth to note that usage of Core PHP is not just about the performance, but also improves the resource usage. In example, according to [2], create action consumes 0.14 Megabytes of memory in Core PHP, 0.17 Megabytes when using Phalcon and 9.42 Megabytes when using Laravel. This is directly connected with the number of called functions during this operation, which is equal to 50 in Core PHP 225 for Phalcon and 1253 for Laravel. The difference between these numbers is significant.

Thus, in the projects that don't require much scalability and are not very complicated we need to omit the usage of the frameworks. Based on the information provided by HTTP Archive [5], the average web page size in 2010 was 702 KB compared to in 2016 which is 2232 KB. With the increase in size of every web component and the addition of video, web page size has increased by 1,530 KB or 317 % from 2010 to 2016. Developers should keep in mind that every request their application consumers do to the server is the electricity consumption as well as working resource of the server devices.

The last but not least point in discussing the Core PHP and PHP Frameworks is the security. This point is where we can summarize the comparison, because the same thing applies to all aspects of web application web development. Everything that is done in terms of security in frameworks, obviously can be implemented in Core PHP. When using the Core PHP, you'll get clear traceable code that uses minimal amount of resources and executes faster, but the whole development process will be significantly more time consuming and complicated than the development process that uses frameworks. Also, the bigger the Core PHP project, the harder it will be to support and develop, especially when working in a team.

So, the conclusions of the comparative analysis are:

1) if a project is large in terms of scope, is expected to be scaled in future and is developed by a team, it's better to use PHP frameworks;

2) if the project is relatively small and simple, amount of customers is known and it is supported by small group of developers or single developer, it's better to use Core PHP to keep the project simple and avoid unnecessary resource consumption.

The main question that remains after these conclusions is how do we keep the advantages of MVC architecture when not using MVC framework. Let's consider the simple example – dictionary application to store and search through translations of words and word combinations between English and Ukrainian languages. The database structure contains the following tables:

– categories (to store translation categories);

– en_lang_parts (to store the names of English language parts);

– invitations (to store the invitations that administrator sends to the potential editors);

– remarks (to store the editions made by editors into existing translations);

– roles (to store user roles);

– translations (to store the translations and it's context);

– ua_lang_parts (to store the names of Ukrainian language parts);

– users (to store the user data.

The sitemap of the project is given on Fig. 1.

The functionality of the project is achieved through the structure shown on Fig. 2.

Root project folder contains two folders (app and assets) and three files:

– config.php file is used to store environment variables like database credentials, project root folder path, project URL etc;

– .htaccess file contains the configuration for Apache. The main purpose of this file in this project

is to map all requests to index.php, no matter what the sub-URL is;

– index.php file reads the config.php file and calls the router, which is located in app folder.

Assets folder contains three folders:

– css – for all CSS files of the project;

– js – for all the scripts;

– img – for all the images in the project.

App folder contains two files and three folders:

– router.php file checks the current url the user is located at and loads proper controller to show the page;

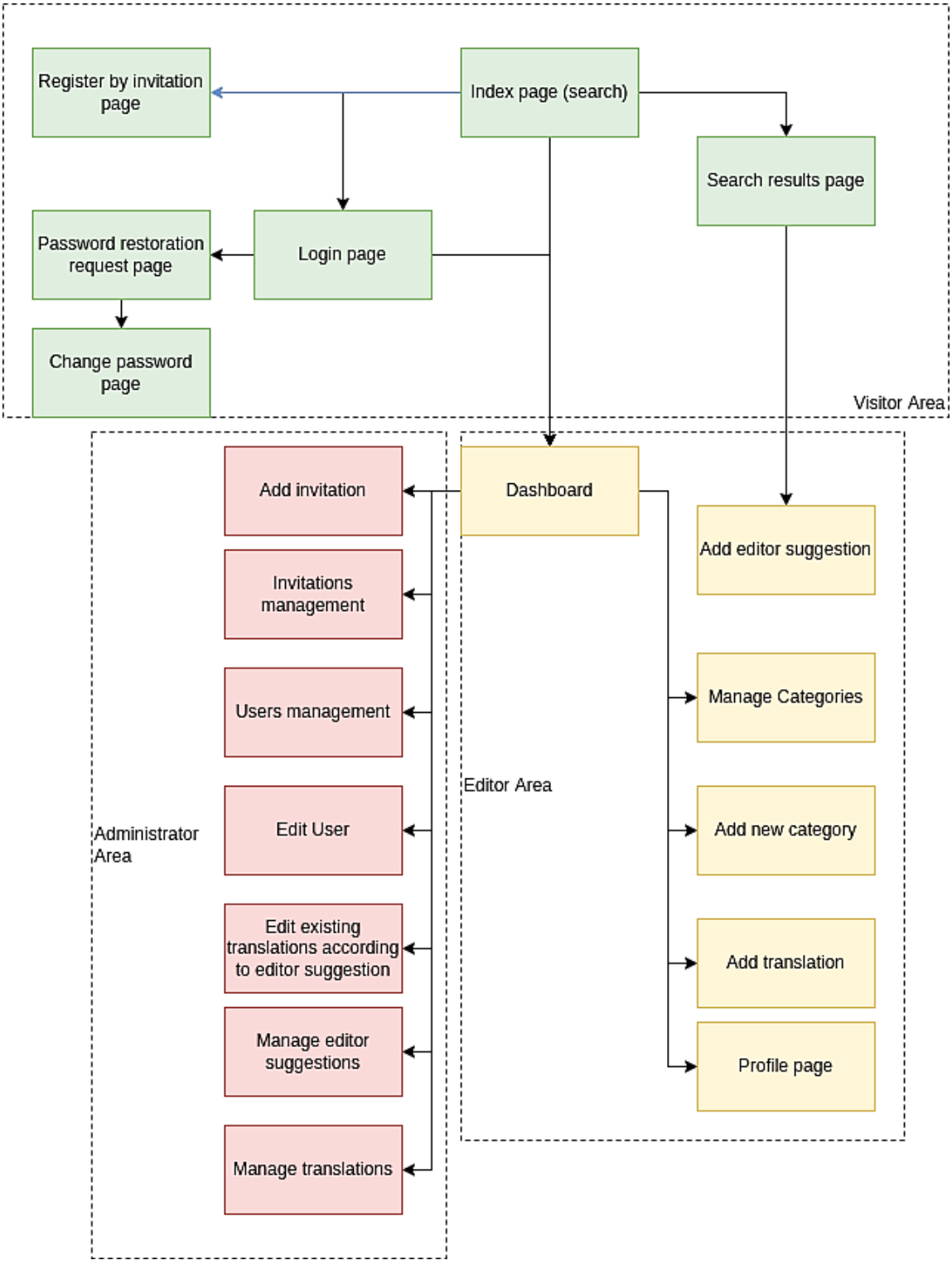

Figure 1 – Project sitemap

– dbconnect.php creates the mysql connection using the credentials so it can be used by models later;

– models folder contains the models for each project entity that are represented with a file that contains the number of functions providing CRUD implementation and additional functionality (like checking the email for availability during registration;

– controllers folder contains the controller files (one for each project page) that call the proper models to provide the data and includes views to create server-side rendering of the result page;

– views folder contains html code fragments for common parts (like header and footer) and specific pages).

This structure is an example of how a simple application can be implemented with Core PHP in an MVC way. Because the whole application is server-side rendered, we can manage access using the PHP sessions. However, if we need to make the app universal for any web clients, it's better to provide an API and use RESTful architecture. For this, we only need to modify the app excluding views and replacing them by generating proper JSON-formatted responses in controllers and control the API calls through the $_SERVER["REQUEST_METHOD"] variable (to retrieve the post type). To authenticate the web client, Bearer token can be used and retrieved through processing of the $_SERVER['HTTP_AUTHORIZATION'] variable.

In conclusion, it is worth to note that cleanness and performance of each application is mostly dependent on developer skills and his ability to simplify the developed apps. Avoiding unnecessary complexity is the key step to build a sustainable web environment, and this is the task of every developer.

Acknowledgements. *Author wants to thank the Armed Forces of Ukraine and all the defenders of Ukraine that give us the possibility to proceed scientific and engineering work in time of war.*

REFERENCES

1. Usage statistics of PHP for websites. (2022). *W3Techs*. Retrieved May 25, 2022, from: https://w3techs.com/technologies/details/pl-php

2. Piróg, W. (2018). *How a PHP framework can speed up the development process*. Polcode. Retrieved May 25, 2022, from: https://polcode.com/resources/blog/how-a-php-framework-can-speed-up-the-development-process/

3. eTatvaSoft. (2016). *Basic difference in Web Development with Core PHP and PHP Frameworks.* eTatvaSoft. Retrieved May 25, 2022, from: https://www.etatvasoft.com/blog/basic-difference-in-web-development-with-core-php-and-php-frameworks/

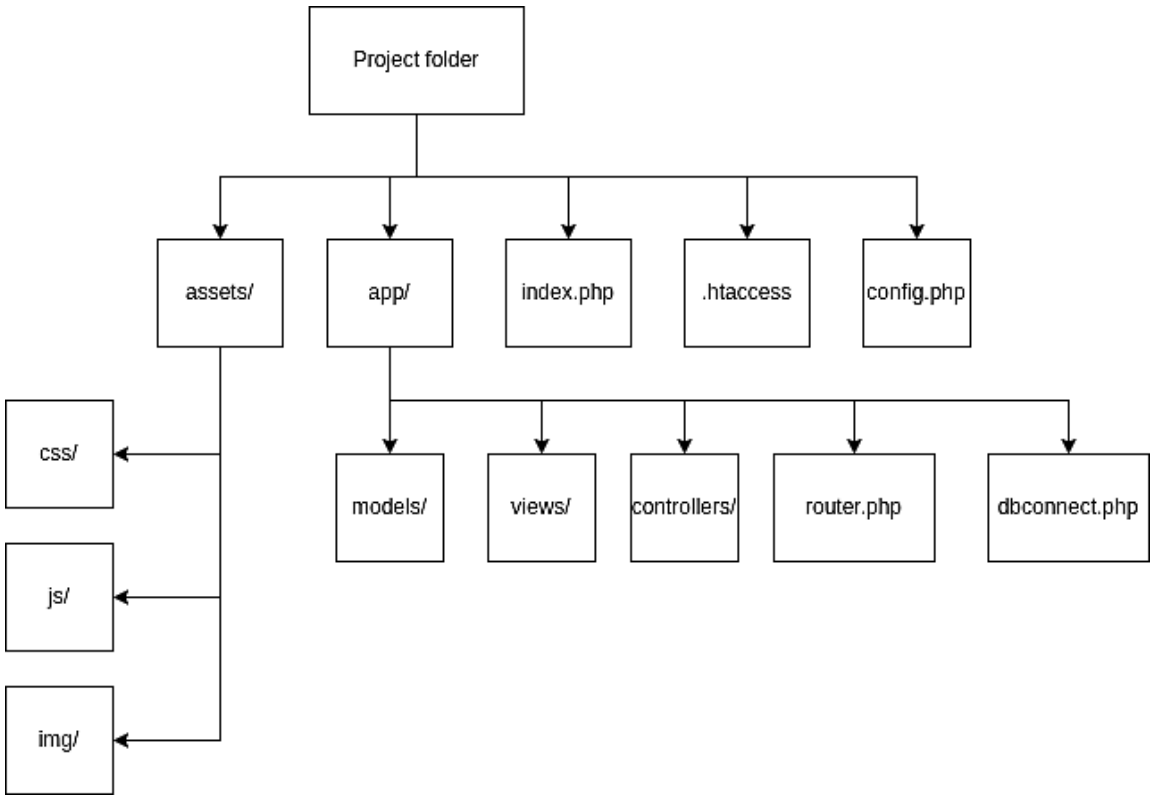

Figure 2 – Project structure

4. Samra, J. (2015). Comparing Performance of Plain PHP and Four of Its Popular Frameworks (Dissertation). Retrieved from: http://urn.kb.se/resolve?urn=urn:nbn:se:lnu:diva-45691

5. KeyCDN. (2018). *The Growth of Web Page Size.* Retrieved May 25, 2022, from: https://www.keycdn.com/support/the-growth-of-web-page-size

CORE PHP ТА PHP ФРЕЙМВОРКИ: ПОРІВНЯЛЬНИЙ АНАЛІЗ ДВОХ ПІДХОДІВ ДО РОЗРОБКИ СЕРВЕРНИХ ДОДАТКІВ

Мар'ян Слабінога
кандидат технічних наук, доцент,
доцент кафедри комп'ютерних систем і мереж
Івано-Франківський національний технічний університет нафти і газу, вул. Карпатська, 15, Івано-Франківськ, Україна, 76019, marian.slabinoha@nung.edu.ua;
ORCID: 0000-0002-7296-0356

Метою даної роботи є порівняння двох підходів розробки серверних додатків з використанням мови програмування PHP – Core PHP («чистого PHP») і PHP фреймворків. Обидва підходи популярні, але, як і в будь-якій іншій мові, використання фреймворків економить час розробника і дозволяє йому не реалізовувати рутину, пов'язану з тривіальними операціями.

Однак, коли ми говоримо про продуктивність і розміри веб-додатка, можна сказати, що використання фреймворків часто включає багато зайвого та непотрібного коду з пакетів, які використовуються. Часто, веб-розробники не приділяють достатньо уваги контролю за цими процесами, в результаті чого розроблені web-додатки стають важчими і споживають більше ресурсів серверних систем. Більше того, підхід з використанням фреймворків часто використовується для дуже простих додатків, які можна легко реалізувати за допомогою Core PHP і на основі кількох файлів коду. Це створює невиправдано великі проекти, що вирішують дрібні та прості завдання.

Мета цієї статті – висвітлити переваги та недоліки веб-розробки PHP з використанням Core PHP та PHP фреймворків. Здебільшого досліджуються безпека та продуктивність, але враховуються й деякі менш критичні речі. Надано рекомендації щодо використання обох підходів, щоб забезпечити найбільш ефективне поєднання досвіду розробки та продуктивності.

В результаті порівняльного аналізу пропонується MVC архітектура додатка, подібна до фреймворку, для створення невеликих легких веб-додатків для користувацьких завдань (у нашому випадку, веб-сайт словник). Надається структура програми та виділено ключові моменти розробки (за допомогою Core PHP).

Обговорюються перспективи модифікації для вирішення тривіальних проблем, таких як серверний додаток RESTful або аутентифікація з допомогою Bearer-токенів.

Ключові слова: веб-розробка в контексті сталого розвитку, оптимізація продуктивності, PHP-фреймворки, архітектура MVC, «чистий» PHP.

ЛІТЕРАТУРА

1. Онлайн ресурс Web Dev : веб-сайт. URL: https://web.dev (дата звернення: 25.05.2022).

2. Онлайн ресурс Polcode. How a PHP framework can speed up the development process : веб-сайт. URL: https://polcode.com/resources/blog/how-a-php-framework-can-speed-up-the-development-process/ (дата звернення: 25.05.2022).

3. Онлайн ресурс eTatvaSoft – Basic difference in Web Development with Core PHP and PHP Frameworks : веб-сайт. URL: https://www.etatvasoft.com/blog/basic-difference-in-web-development-with-core-php-and-php-frameworks/ (дата звернення: 25.05.2022).

4. Samra J. Comparing Performance of Plain PHP and Four of Its Popular Frameworks: Dissertation on Computer Sciences. Linnaeus University, Kalmar, 2015. 33 p.

5. Онлайн ресурс KeyCDN – The Growth of Web Page Size : веб-сайт. URL: https://www.keycdn.com/support/the-growth-of-web-page-size (дата звернення: 25.05.2022).

Стаття надійшла 03.03.2022