JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 11 (2019) 86-90
Wysłane: 2018-11-27
Przyjęte: 2018-12-01

# Tworzenie aplikacji internetowych na platformie JEE i PHP – analiza porównawcza

Sebastian Jędrych*, Bartłomiej Jędruszak, Beata Pańczyk
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

**Streszczenie**. Artykuł analizuje możliwości tworzenia aplikacji internetowych z wykorzystaniem dwóch konkurencyjnych rozwiązań opartych na językach Java (Spring) i PHP (Laravel). Porównano między innymi elementy implementacji, aspekty bezpieczeństwa oraz wydajność stworzonych aplikacji testowych. Celem badań było wskazanie platformy oferującej większe możliwości, określenie która jest łatwiejsza do opanowania i bardziej przyjazna dla programisty, a także wskazanie, która jest bardziej wydajna. Analizując rozwiązania stosowane w obu technologiach – podjęto próbę oceny ich konkurencyjności względem siebie.

**Słowa kluczowe:** PHP; JAVA; Spring; Laravel

*Autor do korespondencji.

Adres e-mail: jedrychsebastian@gmail.com

# Comparative analysis of web applications development using JEE and PHP

Sebastian Jędrych*, Bartłomiej Jędruszak, Beata Pańczyk
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

**Abstract.** The article analyses the possibilities of creating web applications using two competing solutions based on Java (Spring) and PHP (Laravel) languages. The elements of implementation, security aspects and the efficiency of created test applications were compared. The aim of the research was to identify a platform offering greater opportunities, easier to learn and more programmer-friendly, and also a more efficient indication. Analysing the solutions used in both technologies - an attempt was made to assess their competitiveness with each other.

**Keywords:** PHP; JAVA; Spring; Laravel

*Corresponding author.

Adres e-mail: jedrychsebastian@gmail.com

## 1. Wstęp

Liczba dostępnych na rynku rozwiązań do wytwarzania oprogramowania zmusza do ciągłego poszukiwania technologii umożliwiającej szybkie i bezpieczne zrealizowanie projektu. Podejmując decyzję o wyborze języka programowania oraz konkretnego szkieletu programistycznego, czy też konkretnego szablonu programistycznego, najczęściej korzysta się ze źródeł zbierających statystyki i opinie użytkowników, np. TIOBE [1]. Opierając się na danych w nim zawartych, wytypowano Javę i PHP, jako języki cieszące się obecnie popularnością wśród twórców aplikacji internetowych. Dodatkowo skorzystano z najbardziej popularnych frameworków reprezentujących badane technologie – Spring dla Javy (Rys.1) i Laravel dla PHP (Rys. 2).

## 2. Cel badań

Celem porównania było sprawdzenie, który z frameworków jest technologią oferującą większe możliwości wytwarzania oprogramowania i który osiąga lepsze wyniki w testach wydajnościowych.

**Java Web Frameworks Index** | REBELLABS by ZEROTURNAROUND

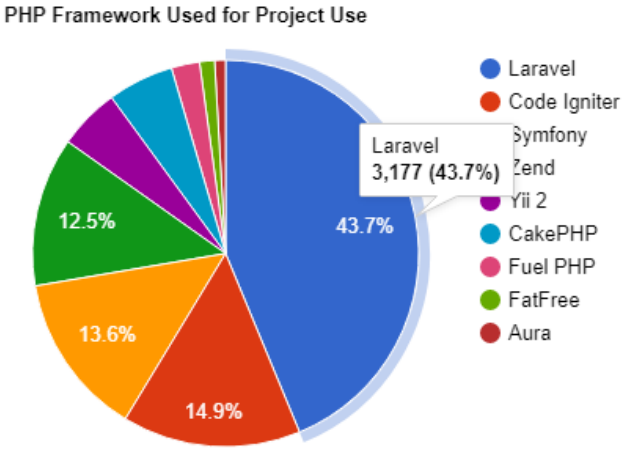
| Rank | Framework | Popularity |
|---|---|---|
| 1 | Spring MVC | 32.00 |
| 2 | JSF | 22.30 |
| 3 | GWT | 9.84 |
| 4 | Spring Boot | 9.45 |
| 5 | Grails | 8.50 |
| 6 | Struts | 7.21 |
| 7 | Play framework | 6.00 |
| 8 | Vaadin | 2.75 |
| 9 | Dropwizard | 1.21 |
| 10 | JHipster | 0.74 |

Rys. 1. Popularność frameworków w roku 2018 dla Javy [2]

## 3. Metoda badawcza

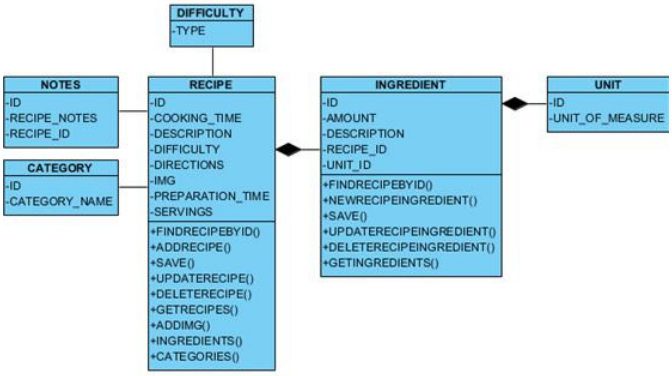
Badania zrealizowano za pomocą dwóch aplikacji testowych, zaimplementowanych w wybranych

frameworkach. Obie aplikacje oferują takie same funkcjonalności prostej aplikacji typu CRUD. W obu, interfejs graficzny jest realizowany z wykorzystaniem szablonów Bootstrap, obie też wykorzystują ten sam system zarządzania danymi MySQL. Z uwagi na odmienną specyfikę technologii do zarządzania szablonami widoków wykorzystano Thymeleaf [4] w projekcie Spring i silnik Blade [5] w projekcie Laravel.




Rys. 2. Popularność frameworków w roku 2018 dla PHP [3]

Aplikacja testowa jest rodzajem książki kucharskiej z przepisami dań i oferuje funkcjonalności typowej aplikacji CRUD. Model bazy danych prezentuje rysunek 3. Interfejs graficzny obu aplikacji wygląda identycznie (Rys. 4).




Rys. 3. Diagram klas wykorzystany w aplikacjach testowych


Rys. 4. Strona główna aplikacji testowej

## 4. Analiza porównawcza

Ogólną charakterystykę wybranych technologii przedstawiono w tabeli 1.

Tabela 1. Ogólna charakterystyka frameworków

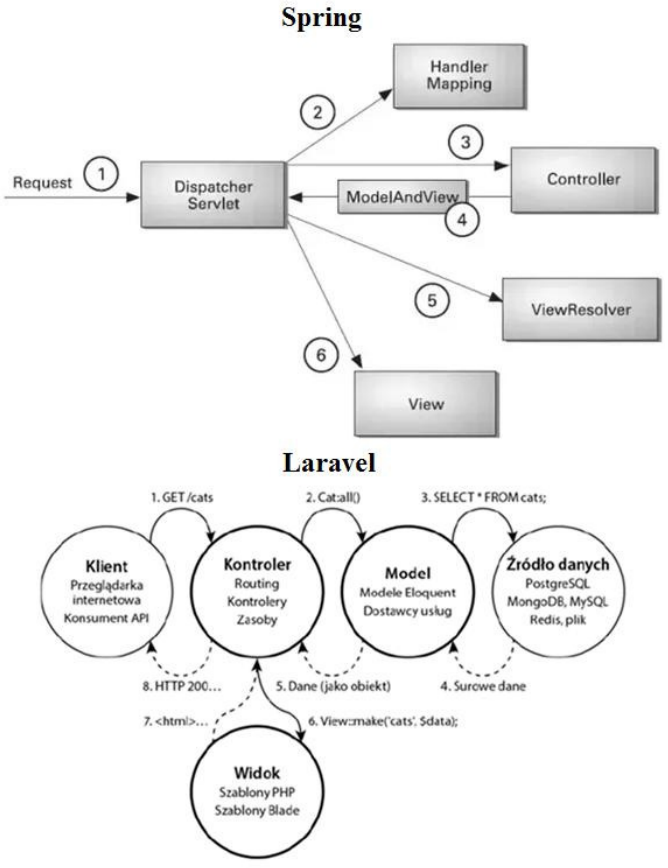
| | **Spring** | **Laravel** |
|---|---|---|
| Twórca | Pivotal Software | Taylor Otwell |
| Data wypadania pierwszej wersji | 1 października 2002 | 9 czerwca 2011 |
| Wersja oraz data wydania najnowszej wersji | 5.1- 21 września 2018 | 5.7- 4 września 2018 |
| Język programowania | Java | PHP |
| .Technologie widoku | Thymeleaf, JSP, jTwig | Blade, Twig, Volt, |
| Wzorzec projektowy | MVC | MVC |
| Licencja | Apache 2.0 | MIT |

Analiza porównawcza została przeprowadzona w oparciu o następujące kryteria:

- model tworzenia aplikacji,
- obsługa operacji bazodanowych,
- bezpieczeństwo,
- wydajność i wybrane metryki kodu.

### 4.1. Model tworzenia aplikacji

Oba frameworki zostały oparte na tym samym szablonie - MVC (ang. Model-View-Controller), jednak każdy realizuje własne modyfikacje, przedstawione na rysunku 5.




Rys. 5. Modyfikacje modelu MVC w Spring i Laravel [6, 7]

Za tworzenie elementów aplikacji w projekcie Laravel odpowiedzialny jest interfejs Artisan CLI, skojarzony z konsolą danego systemu lub zaimplementowany w wybranym IDE. Spring został oparty na metodzie adnotacji, które nadają klasom odpowiednią rolę. Przykład 1 i 2 przedstawia oba te podejścia dla kontrolera, odpowiednio w Spring i Laravel.

Przykład 1. Kontroler z adnotacjami w Spring

```
@Controller
public class Kontroler {
    @GetMapping
    Typ index() {}
    @PostMapping
    Typ save() {}
    @PutMapping(„/{id}”)
    Typ edit(@PathVariable Typ id) {}
    @DeleteMapping(„/{id}”)
    Typ delete(@PathVariable Typ id) {}
}
```

Przykład. 2. Kontroler w Laravel

```
class Kontroler extends Controller
{
    public function index() {}
    public function save() {}
    public function edit($id) {}
    public function delete($id) {}
}
```

### 4.2. Obsługa bazy danych

Liczba standardowych i najczęściej wykonywanych operacji na bazie danych (tzn. operacji typu CRUD) pozwala na stworzenie jednolitego mechanizmu dostępu do danych i ich modyfikacji dla wielu modeli aplikacji. Takie podejście zastosowano w Eloquent ORM [8], który jest podstawowym sposobem obsługi obiektów bazodanowych w Laravel (Przykład 3).

Przykład. 3. Eloquent ORM w Laravel

```
return Notes::where('RECIPE_ID', $this->ID)
    ->select('NOTES.RECIPE_NOTES')
    ->get();
```

Rozszerzoną składnię poleceń SQL w Laravel udostępniono za pośrednictwem Query Builder, którego składnia jest zbliżona do podstawowej składni języka PHP, opierającego się na parametryzacji (Przykład 4).

Przykład. 4. Query Builder w Laravel

```
return DB::table('RECIPE')
    ->Join('NOTES', function($join)
    {
     $join->on('RECIPE.ID', '=', 'NOTES.RECIPE_ID')
          ->where('NOTES.RECIPE_ID', '=', $this->ID);
    })
    ->select('NOTES.RECIPE_NOTES')
    ->get();
```

Spring pozwala obsługiwać bazę danych na wiele sposobów, ale szczególnie przydatne są mechanizmy ORM (np. Hibernate) oraz podstawowy interfejs JDBC. Od wersji Spring 2.0, za standard uważa się Spring Data [9], który opiera się na sygnaturach będących podstawą języka DSL (ang. domain-specific language). Dzięki temu, za pośrednictwem jednego z dostępnych interfejsów, możliwe jest zarządzanie bazą bez znajomości języka SQL (Przykład 5).

Przykład 5. Interfejs JpaRepository w Spring

```
@NoRepositoryBean
public interface JpaRepository<T, ID>
    extends PagingAndSortingRepository<T, ID>,
    QueryByExampleExecutor<T>
{
    List<T> findAll();
    List<T> finAll(Sort varl);
    List<T> findAllById(Iterable<ID> varl);
    <S extends T> List<S> saveAll(Iterable<S> varl);
}
```

### 4.3. Bezpieczeństwo

Zabezpieczenia w aplikacjach internetowych są tematem nie tylko skomplikowanym z naukowego punktu widzenia, ale też trudnym do implementacji z powodu wielu możliwości potencjalnych ataków i nieprzewidywalności działań użytkowników. Technologie starają się nadążać z identyfikowaniem powszechnych mechanizmów wykorzystywanych do uzyskania wrażliwych danych. Obrona przed typowym atakiem SQL Injection, czy zabezpieczenie przesyłanych danych z wykorzystaniem kryptografii, stało się koniecznością. Na rysunku 6 przedstawiono reprezentacje mechanizmu uwierzytelnienia połączenia dla obu technologii.

**Spring**

```
1  @Bean
2  public PasswordEncoder passwordEncoder() {
3      return new BCryptPasswordEncoder();
4  }
```

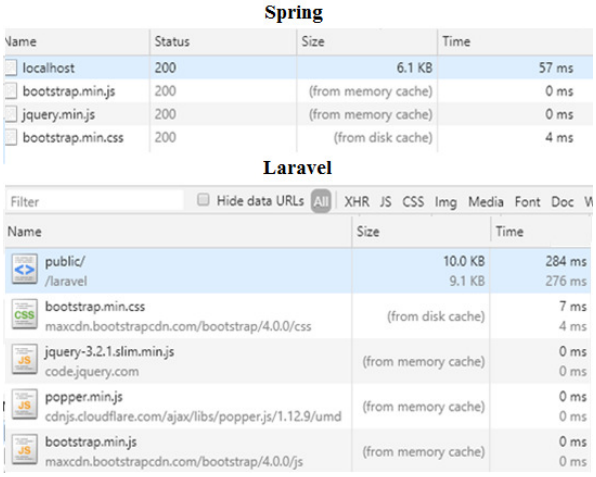
**Laravel**

```
<input type="hidden" name="_token" value="{{ csrf_token() }}">
```

Rys. 6. Uwierzytelnienie połączenia

### 4.4. Wydajność

Specyfika aplikacji niejednokrotnie wymusza efektywne renderowanie zawartości strony. Podejścia do sposobu wyświetlania są ściśle powiązane z platformą, co ma swoje odwzorowanie w czasie, który jest potrzebny do załadowania całej zawartości. Celem zbadania tego kryterium, skorzystano z ChromeDev Tools. Narzędzie to jest wbudowane w przeglądarkę oraz program JMeter, który pozwala symulować obciążenie aplikacji. Na rysunku 7 zestawiono informacje uzyskane z narzędzia deweloperskiego przeglądarki Google Chrome. Porównując czasy odpowiedzi dla aplikacji Spring i Laravel, można dostrzec znaczne różnice, spowodowane wykonywaniem kodu PHP osadzonego w widokach aplikacji. Odpowiedzialny za to jest interpreter PHP, który odczytuje, przetwarza i wykonuje kod, który w przypadku Javy jest dynamicznie implementowany, eliminując konieczność oczekiwania na cały zestaw danych.




Rys. 7. Czasy odpowiedzi zmierzone za pomocą ChromeDev Tools

Pomiary uzyskane za pośrednictwem JMeter, dla jednakowych parametrów obciążenia, przedstawiono w tabelach 2 i 3.

Tabela 2. Czas odpowiedzi dla strony głównej aplikacji

| Numer pomiaru | Czas odpowiedzi Laravel [ms] | Czas odpowiedzi Spring [ms] |
|---|---|---|
| 1 | 900 | 205 |
| 2 | 1050 | 165 |
| 3 | 440 | 150 |
| 4 | 410 | 135 |
| 5 | 480 | 180 |
| 6 | 1250 | 105 |
| 7 | 590 | 195 |
| 8 | 1150 | 135 |
| 9 | 590 | 125 |
| 10 | 810 | 105 |
| **Średni czas dla 10 pomiarów** | **714** | **150** |

Tabela 3. Czas odpowiedzi dla wybranej podstrony

| Numer pomiaru | Czas odpowiedzi Laravel [ms] | Czas odpowiedzi Spring [ms] |
|---|---|---|
| 1 | 520 | 170 |
| 2 | 480 | 135 |
| 3 | 470 | 85 |
| 4 | 780 | 75 |
| 5 | 430 | 105 |
| 6 | 500 | 55 |
| 7 | 420 | 125 |
| 8 | 480 | 60 |
| 9 | 440 | 75 |
| 10 | 510 | 65 |
| **Średni czas dla 10 pomiarów** | **503** | **95** |

### 4.5. Wybrane metryki kodu

Tabela 4 zawiera zestawienie podstawowych metryk kodu. Wyliczenia przeprowadzone zostały za pomocą wtyczki Statistic do IntelliJ IDEA, współpracującej z wieloma technologiami oferującej analizę kodu względem rozszerzeń plików projektu. Wysokie wartości dla Laravel wynikają z wykorzystania bibliotek PHP wchodzących w skład frameworka, które umożliwiają uruchomienie aplikacji testowej bez konieczności instalowania dodatkowych pakietów, które potrzebuje Spring – podstawą jest Java Development Kit, z którego można skorzystać za pośrednictwem jednego z wielu IDE.

Tabela 4. Wybrane metryki kodu

| | Spring | Laravel |
|---|---|---|
| Rozmiar projektu [MB] | 1,8 | 49 |
| Liczba plików | 207 | 21293 |
| Liczba linii kodu HTML | 646 | 2181 |
| Liczba linii kodu Java/PHP | 1208 | 444025 |

## 5. Analiza wyników

Implementacja aplikacji testowych oraz przeprowadzona analiza porównawcza, posłużyły do opracowania tabeli, prezentującej ocenę autorów dla obu frameworków w badanych kategoriach. Tabela 4 przedstawia wyniki w skali 1-5, gdzie 5 jest oceną najwyższą, a 1 najniższą. W kryterium dotyczącym obsługiwanych baz danych, za każdą obsługiwaną bazę przyznawany jest 1 punkt. W przypadku kryterium opisującego stopień złożoności danego frameworka – wartość 5 oznacza bardzo łatwy (wymagana niewielka wiedza deweloperska), natomiast wartość 1 oznacza bardzo trudny (wymagana duża wiedza i doświadczenie deweloperskie).

Tabela 5. Końcowa ocena frameworków

| Kryterium oceny | Laravel | Spring |
|---|---|---|
| Przejrzystość tworzenia modelu aplikacji | 4 | 4 |
| Poziom trudności tworzenia aplikacji | 4 | 3 |
| Obsługiwane bazy danych | 5 | 5 |
| Dostępność materiałów | 4 | 5 |
| Wsparcie społeczności | 5 | 5 |
| Wymagany poziom znajomości frameworka | 5 | 2 |
| Przejrzystość struktury katalogów aplikacji | 4 | 5 |
| Wydajność wyświetlania rekordów | 2 | 5 |
| **Końcowa ocena punktowa** | **32** | **33** |

## 6. Wnioski

Na podstawie przeprowadzonych badań można sformułować następujące wnioski:

- Laravel należy uznać za platformę prostszą do nauki w porównaniu do Spring;
- poziom abstrakcji w Laravel jest na wyższym poziomie niż w Spring;
- liczba oferowanych pakietów i funkcji tworzy z frameworka Spring narzędzie do budowy zaawansowanych aplikacji, nie tylko internetowych;
- Laravel bardziej nadaje się do tworzenia małych i średnich projektów;
- do zadań, które wymagają dużej wydajności czasowej zalecany jest Spring;
- liczba dostępnych materiałów i poradników dla Spring jest znacznie obszerniejsza niż w przypadku Laravel.

## Literatura


[1] Statystyki popularności TIOBE, https://www.tiobe.com/tiobe-index/ [25.11.2018]

[2] Popularność frameworków Javy, https://zeroturnaround.com/rebellabs/java-web-frameworks-index-by-rebellabs/ [26.11.2018]

[3] Popularność frameworków PHP, https://coderseye.com/best-php-frameworks-for-web-developers/ [27.11.2018]

[4] Thymealeaf, https://www.thymeleaf.org/ [25.11.2018]

[5] Blade, https://laravel.com/docs/5.6/blade [25.11.2018]

[6] Walls C.: Spring w akcji. Wydanie IV, HELION, Gliwice, 2015.

[7] Saunier R.: Laravel 4 Podstawy tworzenia aplikacji w PHP, HELION, Gliwice, 2015

[8] Eloquent ORM, https://laravel.com/docs/5.7/eloquent [25.11.2018]

[9] Spring Data JPA, https://docs.spring.io/spring-data/jpa/docs/current/reference/html/ [25.11.2018]