JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 6 (2018) 1-5
Wysłane: 2017-09-13
Przyjęte: 2017-09-19

Porównanie technologii tworzenia aplikacji internetowych na przykładzie Spring MVC i Vaadin

Łukasz Tomczyk*, Beata Pańczyk
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przedstawiono wyniki porównania efektywności wytwarzania aplikacji internetowych na platformie Java w wersji 1.8 z zastosowaniem szkieletów programistycznych Spring MVC oraz Vaadin. Analiza porównawcza została przeprowadzona za pomocą autorskich aplikacji testowych, implementujących te same funkcjonalności w obu technologiach.

Słowa kluczowe: Spring MVC; Vaadin; aplikacje internetowe; framework

*Autor do korespondencji.

Adres e-mail: lukasz.tomczyk1@pollub.edu.pl

Comparison of web applications development using Spring MVC and Vaadin

Łukasz Tomczyk*, Beata Pańczyk
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents the results of web application development effectiveness on Java platform in 1.8 version using Spring MVC and Vaadin frameworks. The comparative analysis was conducted with test applications, implementing the same functionalities in both technologies.

Keywords: Spring MVC, Vaadin, web applications, framework

*Corresponding author.

E-mail address: lukasz.tomczyk1@pollub.edu.pl

1. Wstęp

Istnieje wiele różnych technologii, wykorzystywanych do tworzenia aplikacji internetowych w środowisku języka Java [1]. Można je podzielić na trzy grupy:

- koncentrujące się tylko na wizualnej stronie interfejsu użytkownika;
- sterujące całą logiką aplikacji i procesami biznesowymi;
- pozwalające na implementację strony wizualnej i logiki aplikacji.

Każde z tych podejść posiada swoje wady i zalety i bardzo trudno jest wskazać najlepsze rozwiązania.

W artykule przeanalizowano dwa wybrane szkielety programistyczne: najbardziej popularny Spring MVC [1] oraz cieszące się coraz większą popularnością Vaadin (Rys. 1).

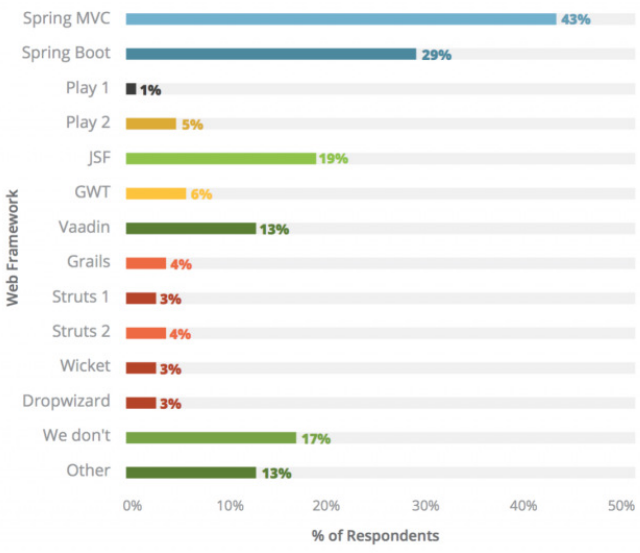
2. Cel, teza i metody badań

Celem badań było porównanie kodu oraz przetestowanie efektywności działania dwóch aplikacji (o takiej samej funkcjonalności) utworzonych za pomocą Spring MVC i Vaadin.

W artykule postawiono następującą tezę:

Framework Vaadin jest bardziej efektywnym narzędziem wytwarzania aplikacji internetowych w porównaniu do Spring MVC.

Dla potwierdzenia przedstawionej tezy wykorzystano metodę badań opartą na analizie porównawczej obu rozwiązań.



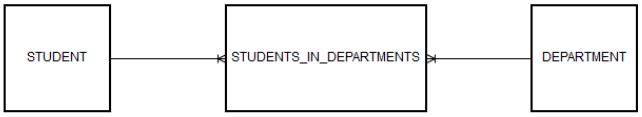
Rys. 1. Popularność wybranych frameworków Java w roku 2016 [2]

3. Aplikacje testowe

Aplikacje Spring MVC oraz Vaadin:

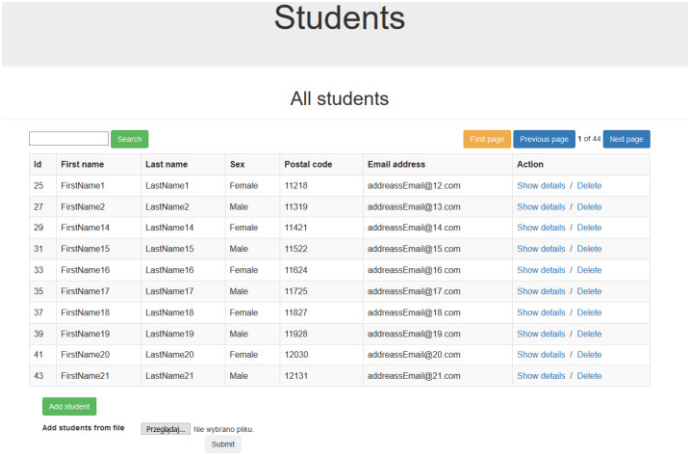
- stworzono w środowisku programistycznym Eclipse [3] z wykorzystaniem narzędzia do automatyzacji i budowy projektów Maven;
- współpracują z serwerem bazy danych Oracle [4], realizując dostęp do danych za pośrednictwem Hibernate;
- wykorzystują funkcje biblioteki jQuery.

Obie aplikacje testowe posiadają typowe funkcjonalności CRUD (ang. Create, Read, Update, Delete, Search), czyli tworzenie, edycję oraz usuwanie informacji o studentach z bazy danych (Rys. 2).



Rys. 2. Schemat bazy danych

Interfejs obu aplikacji jest bardzo podobny oraz umożliwia pełny dostęp do wszystkich funkcjonalności. Został on przedstawiony na rysunku 3.



Rys. 3. Przykładowa strona aplikacji testowej

4. Analiza porównawcza

Najważniejsze cechy Spring MVC i Vaadin zostały zestawione w tabeli 1. W tabeli 2 przedstawiono parametry sprzętowe komputera, na którym wykonano testy.

Do przeprowadzenia porównania obu aplikacji wykorzystane zostały następujące kryteria:

- ogólna struktura aplikacji;
- efektywność pracy z danymi;
- podstawowe metryki kodu;
- efektywność wczytywania zasobów aplikacji przez przeglądarkę internetową.

Tabela 1. Najważniejsze cechy Spring MVC i Vaadin

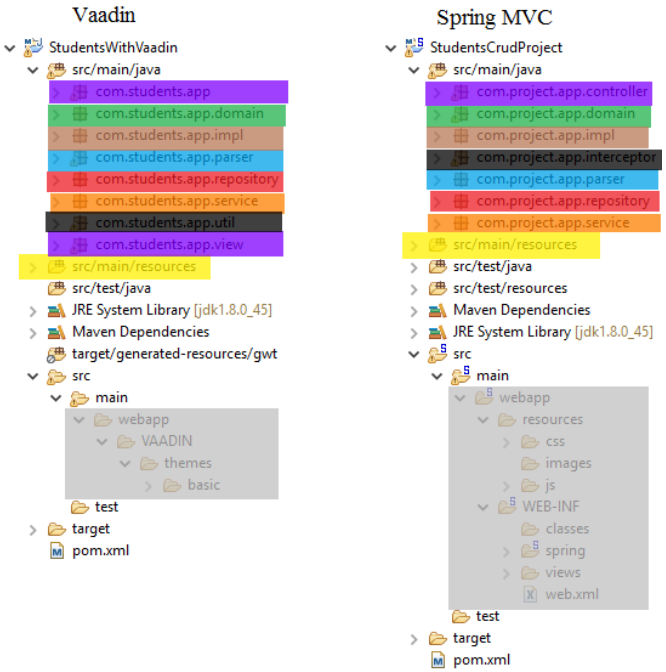
	Spring MVC	**Vaadin**
Twórca	Pivotal Software	Vaadin Ltd.
Pierwsza wersja	1 października 2002	Millstone 3 open source Web framework 2002
Wersja finalna	1 marca 2017 (4.3.7)	22 lutego 2017 (8.1)
Język programowania	Java	
SDK	Java SE 8 SDK	
Technologia widoku	JSP	Java
Inne możliwe technologie widoku	Thymeleaf, Groovy Markup Templates	Html
Serwery open-source	Apache Tomcat, GlassFish, Jetty	
Wsparcie	Bogate wsparcie i dokumentacja, duża baza użytkowników	
Licencja	Apache License 2.0	
Niezależność platformy systemowej	Windows, Mac, Linux	

Tabela 2. Parametry zestawu pomiarowego

Element	**Stan/Wersja**
System operacyjny	Windows 10 64-bitowy
Procesor	Intel Core i7-4712 (2.3-3.3 GHz)
Pamięć RAM	12 GB
Wersja Java	Java 8

4.1. Struktura aplikacji

Rys. 4 przedstawia strukturę obu aplikacji, opartych na projekcie Maven.



Rys. 4. Struktura projektu Spring MVC i Vaadin

Na rysunku wyróżniono kolorem elementy funkcjonalnie wspólne dla obu aplikacji:

- Fioletowy - pakiety klas odpowiadających za realizację żądań oraz interakcji z użytkownikiem;

- zielony - pakiety klas domenowych mapujących encje bazy danych za pomocą adnotacji JPA;
- brązowy - pakiety implementujące interfejsy warstw usług oraz repozytorium;
- niebieski - pakiet z parserem plików txt i csv;
- czerwony - pakiety warstwy repozytorium;
- pomarańczowy - pakiety warstwy usług;
- czarny - pakiety zawierające mechanizmy pomiaru czasów;
- żółty - foldery z plikami konfiguracyjnymi, m.in. połączenia z bazą danych;
- szary - dodatkowe pliki (css, javascript, jsp, xml).

Dzięki zastosowaniu struktury Maven, elementy obu aplikacji są zbliżone. Główne różnice to pliki jsp, xml oraz javascript, które zawiera Spring MVC.

4.2. Efektywność pracy z bazą danych

W celu porównania efektywności obu aplikacji, zmierzono czas pracy z danymi. Do testu przygotowano 9 scenariuszy testowych (Tabela 3.). Każdy scenariusz powtórzono 10 razy a wyniki pomiarów uśredniono.

Tabela 3. Tabela 3. Scenariusze testowe

Scenariusz	Opis
1	Zapis 10 studentów
2	Odczyt 10 studentów
3	Zapis 100 studentów
4	Odczyt 100 studentów
5	Zapis 1000 studentów
6	Odczyt 1000 studentów
7	Zapis 10000 studentów
8	Odczyt 10000 studentów
9	Wyszukiwanie studentów losowo wybranymi słowami kluczowymi

Testowy zbiór studentów został wygenerowany przez prosty program własnego autorstwa. Czas mierzono w milisekundach [ms]. Operacje na bazie realizowano za pomocą Hibernate.

Fragment kodu z pomiarem czasu operacji dla scenariusza 1 oraz 2 pokazują odpowiednio przykłady 1 i 2. W pozostałych scenariuszach pomiar odbywał się analogicznie.

Przykład 1. Pomiar czasu zapisu danych – Spring MVC

```
@RequestMapping(value = "/" +
StudentsAdminViewName,
    method = RequestMethod.POST)
    private String addStudentsFromFile(
        @ModelAttribute("myFile")
        @Valid MyFile myFile,
        HttpServletRequest request) {
     calculator.begin();
     MultipartFile multipartFile =   myFile.getData();
     List<Student> st =
         TextFileParser.getStudentsFromFile(multipartFile);
     studentsService.addStudents(st);
     calculator.end();
     calculator.calculate();
     students =   studentsService.getStudents(
         elementsInPage, startPosition, Order.ASC);
     studentsService.getAllStudents();
     return "redirect:/" +     StudentsAdminViewName;
}
```

Przykład 2. Pomiar czasu odczytu danych – Vaadin

```
private void updateList() {
    performenceCalculator.begin();
    if (filter == null || filter.isEmpty())
    {    students =   service.getStudents(elementsInPage,
         startPosition, Order.ASC);
     }   else     {
         students =   service.getStudentsByFirstName(filter,
         elementsInPage, startPosition, Order.ASC);
    }
    grid.setItems(students);
    performenceCalculator.end();
    performenceCalculator.calculate();
}
```

W tabelach 4 i 5 przedstawiono odpowiednio rezultaty pomiarów dodawania i odczytu 10 studentów.

Tabela 4. Czasy dodawania jednocześnie 10 studentów.

	Spring MVC	Vaadin
Nr pomiaru	**Czas [ms]**	
1	89	111
2	61	83
3	72	87
4	88	93
5	65	88
6	74	87
7	70	89
8	63	90
9	76	84
10	80	87
Średni czas	**73**	**88**

Tabela 5. Tabela 5. Czasy odczytu 10 studentów

	Spring MVC	Vaadin
Nr pomiaru	**Czas [ms]**	
1	309	9
2	327	10
3	302	14
4	304	7
5	306	8
6	311	11
7	301	13
8	309	12
9	317	8
10	326	9
Średni czas	**313**	**11**

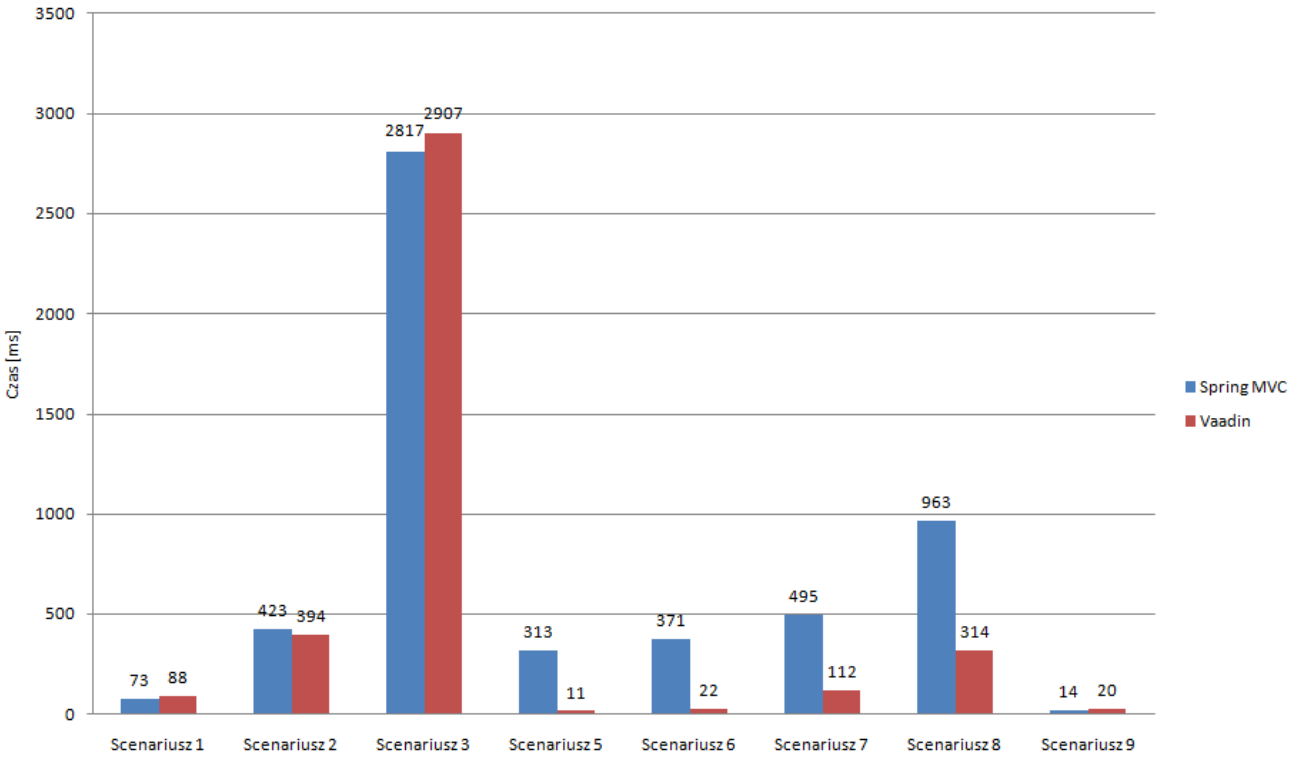
Na rysunku 5a i 5b zestawiono średnie czasy pomiarów dla wszystkich scenariuszy.

Porównując obie aplikacje, w kategorii pobierania rekordów z bazy, Vaadin jest zdecydowanie szybszy od Spring MVC.

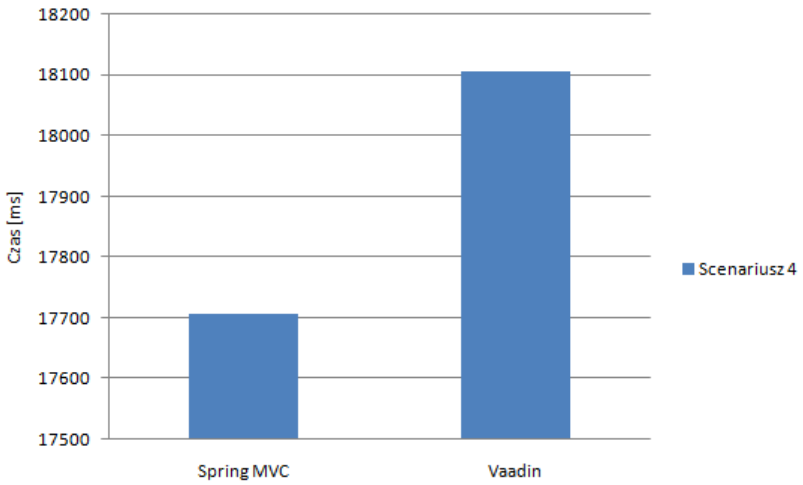
W przypadku zapisu danych, Spring MVC jest szybszy dla małej liczby, natomiast Vaadin dla większej liczby danych.

W przypadku wyszukiwania (scenariusz 9) Spring MVC jest szybszy od Vaadin średnio o 6 milisekund.

.



Rys. 5a. Średnie czasy operacji dla wszystkich scenariuszy testowych dla Spring MVC i Vaadin



Rys. 5b. Średnie czasy operacji dla wszystkich scenariuszy testowych dla Spring MVC i Vaadin

4.3. Efektywność wczytywania zasobów aplikacji w przeglądarce internetowej

Następnym ważnym pomiarem wydajności obu aplikacji jest czas od momentu wystąpienia żądania użytkownika. Czas ten został zmierzony za pomocą konsoli deweloperskiej Mozilla Firefox dla scenariusza 4. Wyniki pomiarów zostały przedstawione w tabeli 6. Na podstawie prezentowanych tam wyników można stwierdzić, że aplikacja Vaadin jest znacznie szybsza niż Spring MVC. Jest to spowodowane rozdzieleniem w Spring MVC warstwy prezentacji od warstwy kontrolera.

W Vaadin obie warstwy znajdują się w jednym miejscu w kodzie źródłowym.

4.4. Porównanie metryk aplikacji.

Ostatnim elementem porównania obu aplikacji jest analiza metryk kodu. Do tego celu wykorzystano narzędzie dostępne w Eclipse. Wyniki przedstawia tabela 7.

Generalnie liczba linii kodu dla Spring MVC jest mniejsza od aplikacji napisanej w Vaadin. Jednak aplikacja w Spring

MVC zajmuje więcej pamięci dyskowej niż w przypadku Vaadin, ponieważ zawiera w sobie więcej wbudowanych mechanizmów.

Tabela 6. Czas pobierania zasobów przez przeglądarkę Mozilla Firefox

	Spring MVC	Vaadin
Nr pomiaru	**Czas [ms]**	
1	568	39
2	602	21
3	589	22
4	563	34
5	612	32
6	538	27
7	513	29
8	558	33
9	503	28
10	613	31
Średni czas	**566**	**30**

Tabela 7. Wybrane metryki kodu obu aplikacji.

	Liczba linii kodu	
	Spring MVC	**Vaadin**
Kod Java	1165	2081
Kod JSP	564	0
Kod JavaScript	57	0
Pliki konfiguracyjne	327	246
Razem linii kodu	2113	2327
Liczba bibliotek	43	36
Rozmiar projektu (MB)	**61,8**	**48,4**

5. Wnioski

Na podstawie przeprowadzonych badań można sformułować następujące wnioski:

- zastosowanie obydwu frameworków do tworzenia aplikacji internetowych wymaga znajomości podstaw języka Java;
- Spring jest stabilną platformą deweloperską, która stale się rozwija i oferuje bogate możliwości przystosowania do różnych projektów [5];
- patrząc na wykresy popularności frameworków, Spring MVC jest wykorzystywany najczęściej. Wiąże się z tym fakt, że wiele projektów powstało dużo wcześniej niż same frameworki [6]. Aplikacje napisane w Springu nadal są w nim utrzymywane;
- Vaadin powstał głównie do tworzenia wyglądu. Dostarcza API, które jest dużym ułatwieniem dla programistów [7];
- biorąc pod uwagę efekt końcowy, czyli działanie aplikacji, użytkownik nie powinien odczuć znacznej różnicy w szybkości działania. Jedyną rzeczą, która może być widoczna, to podobieństwo wszystkich aplikacji pisanych z wykorzystaniem frameworka Vaadin;
- Vaadin posiada dużą pulę standardowych komponentów, które w zupełności wystarczają nawet w przypadku tworzenia zaawansowanych aplikacji. Pisanie własnych komponentów zdarza się bardzo rzadko, o czym można przeczytać w dokumentacji frameworka [7];
- Vaadin jest znacznie szybszy jeśli chodzi o wyszukiwanie danych. Jest to spowodowane asynchronicznym wykonywaniem zadań za pomocą *AJAX*. W tym scenariuszu testowym framework Spring MVC korzysta z tradycyjnych żądań, co ma wpływ na czas reakcji;
- odczyt danych jest również mocniejszą stroną Vaadin. Jest to związane z brakiem separacji warstw aplikacji. Widok jest implementowany w kodzie Javy, podczas gdy w Spring MVC tworzy się oddzielne pliki (np. jsp) oraz w dodatkowym pliku konfiguruje się dopasowanie pliku widoku do klasy kontrolera;

Jeśli chodzi o inne artykuły porównujące framework Spring MVC oraz Vaadin, istnieją w dużo mniej szczegółowej formie. Na portalu internetowym [8] można znaleźć porównanie tych dwóch frameworków. Przedstawia ono tylko opis słowny i ocenę użytkowników – nie zawiera porównania wydajnościowego w postaci tabel i wykresów.

Literatura

[1] G. Amuthan, Spring MVC. Przewodnik dla początkujących, Helion, Gliwice, 2015.

[2] https://zeroturnaround.com/rebellabs/most-popular-java-frameworks-tools-and-libraries-2016/ [09.04.2017]

[3] https://www.eclipse.org/ [07.05.2017]

[4] N. Greenberg, Oracle Database 10g: SQLFundamentals I, Oracle, 2004.

[5] Gutierrez F.: Wprowadzenie do Spring Framework dla programistów Java, Helion, Gliwice 2015

[6] https://spring.io/guides, [05.03.2017]

[7] https://vaadin.com/docs/, [02.04.2017]

[8] https://zeroturnaround.com/rebellabs/the-curious-coders-java-web-frameworks-comparison-spring-mvc-grails-vaadin-gwt-wicket-play-struts-and-jsf/2/