JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 14 (2020) 78-81
Wysłane: 2019-10-17
Przyjęte: 2019-10-23

Analiza porównawcza technologii widoków dla aplikacji Spring

Vadym Borys*, Roman Slezenko*, Beata Pańczyk

Politechnika Lubelska, Katedra Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem artykułu jest wybór, możliwie najbardziej wydajnych narzędzi do tworzenia interfejsu użytkownika dla aplikacji Spring.W artykuleprzeprowadzono porównanie dla 4 wybranych technologii widoków: JSP, Thymeleaf, Wicket i Angular. Testy wydajności czasowej i pamięciowej zostały przeprowadzone z wykorzystaniem Rest API w Spring. Wyniki testów pozwoliły wskazaćnajlepsze rozwiązania.

Słowa kluczowe: wydajność; Spring; Angular; JSP; Thymeleaf; Wicket

*Autor do korespondencji.

Adres e-mail: bublik.drdrdr@gmail.com

Comparative analysis of view technologies for the Spring application

Vadym Borys*, Roman Slezenko*, Beata Pańczyk

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The goal of the article is to choose the most efficient user interface creation tools possible for Spring. The study compares 4 selected view technologies: JSP, Thymeleaf, Wicket and Angular. Time and memory performance tests were carried out using Rest API in Spring. Test results allowed to identify the best solutions.

Keywords: performance; Spring; Angular; JSP; Thymeleaf; Wicket

*Corresponding author.

E-mail address: bublik.drdrdr@gmail.com

1. Wstęp

Oprócz takich kryteriów jak bezpieczeństwo, czytelność kodu, łatwość konfiguracji oraz rozwoju aplikacji, bardzo ważną cechą każdej wykorzystywanej technologii jest wydajność. Każda technologia typu front-end służy do wygenerowania strony internetowej, przesłaniu danych do przeglądarki oraz wyświetleniu treści po stronie użytkownika. Różne technologie mają różny wpływ na obciążenie procesora i wykorzystywanie pamięci operacyjnej (zarówno po stronie serwera, jak i po stronie klienta).

REST (ang. Representational State Transfer) - to styl architektury oprogramowania dla różnych systemów rozproszonych, takich jak WWW, który jest zwykle używany do budowania usług sieciowych [1]. Najważniejszą zaletą serwisu REST jest to, że każdy system może z nim współpracować. Spring z kolei to najpopularniejszy szkielet aplikacji w języku Java, który pozwala również na szybkie tworzenie wysokiej jakości kodu nowoczesnych aplikacji webowych [2, 3]. Do pomiaru obciążenia procesora oraz innych komponentów sprzętowych służy Spring BootActuator, który otwiera punkty wejściowe (ang. endpoints) dla komunikacji za pomocą REST i pozwala na zdalne sprawdzenie stanu uruchomionej aplikacji oraz jej statystyk [4].

Tematem podjętych badań jest porównanie narzędzi do tworzenia widoków (typu front-end), stosowanych dla aplikacji Spring. W artykule zostaną przedstawione cztery technologie do tworzenia interfejsu graficznego: JSP [5] i Wicket[6] oraz nowszy Thymeleaf[7] i Angular [8].Podobne badania były prezentowane m.in. w artykułach [9, 10], gdzie porównywano jedynie po dwa narzędzia. W literaturze nie znaleziono porównań dla opisanych w niniejszym artykule czterech technologii.

2. Cel badań

Celem badań jest wskazanie najbardziej wydajnych technologii tworzenia widoków współpracujących z Rest API w Spring.

Postawiono następujące tezy badawcze:

Angular jest najlepszą, pod względem wydajności czasowej,przy małym wykorzystaniu pamięci RAM, technologią do tworzenia widoku dla Rest API w Spring.

Thymeleaf jest bardzo dobrym narzędziem do tworzenia małych serwisów.

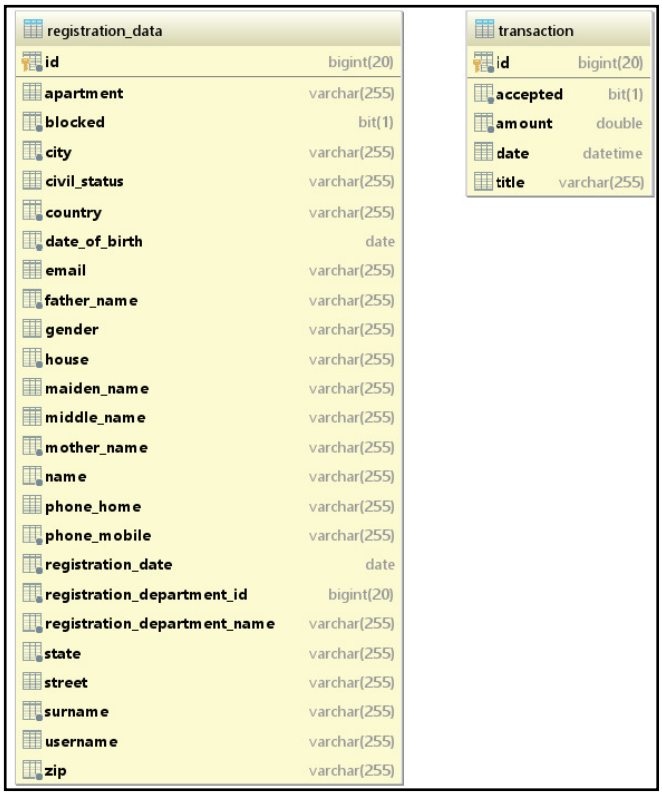
3. Metoda badań

Analizę wydajności czasowej i obciążenia pamięci wykonano za pomocą aplikacji testowej, dla której zaimplementowano cztery różne technologie generowania widoków.

Testowana aplikacja po stronie serwera (back-end) spełnia wymagania pozwalające na wykorzystanie różnych elementów interfejsu użytkownika, którymi są listy, rozbudowane formularze z walidacją oraz duże tabele do prezentacji raportów. Serwerowa część aplikacji testowej (wykonana w Spring,jako część wspólna dla wszystkich testowanych technologii widoku, zapewnia podstawowe mechanizmy dla komunikacji, autoryzacji oraz możliwości dynamicznego ładowania stron.

3.1. Aplikacja testowa

W celu symulowania realistycznych warunków działania aplikacji wykorzystano bazę danych MySQL [11] oraz bibliotekę Hibernate [12], która automatycznie generuje tabele i relacje między nimi na podstawie klas POJO oraz pozwala zarządzać danymi w oparciu o obiekty tych klas. Struktura bazy danych jest przedstawiona na rys. 1. Dla ułatwienia przeprowadzenia badań zostały stworzone klasy pomocnicze oraz konfiguracyjne. Przykład 1 przedstawia kod konfiguracyjny klasy GenerationExecutor, który podczas uruchamiania aplikacji generuje odpowiedni zbiór danych, wykorzystany później w celu wyświetlenia wybranych informacji na stronie. Do pomiaru zajętości pamięci RAM przez aplikację po stronie serwera wykorzystano Spring Actuator i klasę MemoryReader (przykład 2).



Rys. 1. Struktura bazy danych dla aplikacji testowej

Przykład 1. Kod klasy GenerationExecutor generującej przykładowe dane

```
@Service
@RequiredArgsConstructor
public class GenerationExecutor {
    @Value("${data.generate-on-start}")
    private booleangenerationEnabled;
    private final
RegistrationDataGeneratorregistrationDataGenerator;
    private final
TransactionGeneratortransactionGenerator;
    @PostConstruct
    public void generateData() {
        if (generationEnabled) {
        //liczba użytkowników
            registrationDataGenerator.generate(50);
        //liczba transakcji
            transactionGenerator.generate(20000);
        }}}
```

Przykład 2. Kod klasy MemoryReader monitorującej wykorzystywanie pamięci RAM przez aplikację back-end

```
@Service
@RequiredArgsConstructor
public class MemoryReader {

    private final RestTemplate restTemplate;

    public long getMemory(String property) {
        return
(long)(((Double)((Map)((List)this.restTemplate
.getForObject("http://localhost:8080/actuator/metrics/"
+ property, Map.class).get("measurements"))
                .get(0)).get("value"))/(1024*1024));
}
    @Scheduled(fixedRate = 1000)
    public void printMemory() {
System.out.println(getMemory("jvm.memory.used"));
}
}
```

3.2. Konfiguracja środowiska

Badania przeprowadzono na systemach Windows 10 oraz Mac OS Mojave 10.14. Aplikacja back-end działała na serwerzeTomcat.

Parametry komputerów, na których przeprowadzono badania,zostały przedstawione w tabeli 1.

Tabela 1. Parametry komputerów wykorzystanych do badań

Komputer 1	Komputer 2
• Windows 10 Pro 64-bit, • processor: Intel(R) Core(TM) i7 2.6(3.48)GHz, • pamięć RAM: 8 GB, • dysk SSD: 256 GB.	• Mac OS Mojave 10.14, • processor: Intel(R) Core(TM) i7 2,7GHz, • pamięć RAM: LPDDR3 16 GB, • dysk SSD: 512 GB.

3.3. Kryteria i scenariusze badawcze

Wybrane technologie zbadano według następujących kryteriów:

- czas ładowania strony w przeglądarce internetowej;
- obciążenie procesora po stronie serwera;
- wykorzystywanie pamięci RAM;
- inne cechy technologii: dostępność dokumentacji i przykładów, debugowanie, ograniczenia i braki w funkcjonalnościach.

Badania wykonano w oparciu o następujące scenariusze:

- S1 - załadowanie strony zawierającej formularz do tworzenia użytkownika, składający się z dużej liczby różnych elementów, ale bez załadowania danych formularza;
- S2 - wyświetlenie listy elementów w postaci tabeli (mała liczba wierszy);
- S3 - wyświetlenie listy powtarzających się elementów z różnymi danymi w postaci tabeli (10000 wierszy). Test głównie służy do wyodrębniania technologii, które najbardziej efektywnie wykorzystuje przepustowość połączenia inajszybszego wyświetlenia strony.
- S4 - przejście na stronę edycji użytkownika, aktualizacja danych w formularzu i przesłanie zmienionych danych do serwera;

- S5 - przełączenie pomiędzy stronami z małą tabelą w jednej sesji bez czyszczenia pamięci podręcznej a w przypadku testowania technologii Angular, bez przeładowania strony;
- S6 - scenariusz 5 z wykorzystaniem 10000 wierszy w tabeli;
- S7 - przejście na stronę edycji użytkownika, zmiana danych w formularzu i jego zatwierdzenie w jednej sesji.

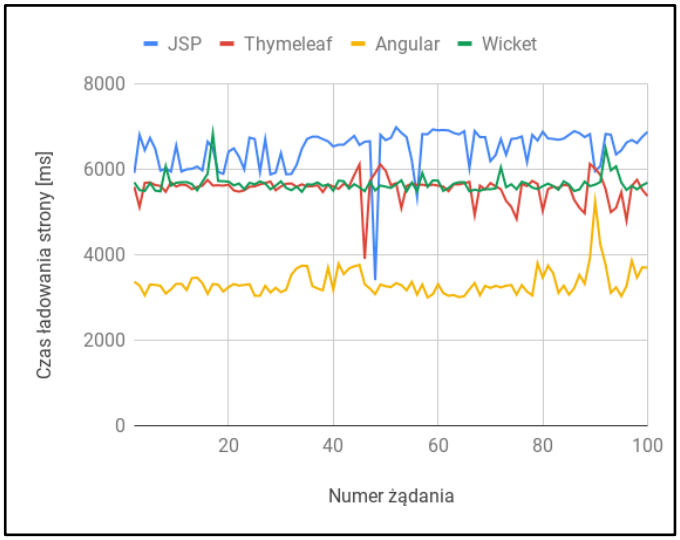
Przed każdym pomiarem dla scenariuszy 1-4 była czyszczona pamięć podręczna, pliki ciasteczek a następnie uruchamiano ponownie emulator przeglądarki. Wszystkie pomiary powtarzano po 100 razy.

Dla testowania front-endu wykorzystano narzędzie developerskie Selenium, pozwalające w łatwy sposób napisać i skonfigurować testy automatyczne, symulujące różne aktywności użytkownika. Jedna z kluczowych zalet Selenium jest obsługa wykonania testów automatycznych na różnych przeglądarkach internetowych.

4. Wyniki badań

4.1. Wydajność czasowa

Różnicę w wydajności badanych technologii najlepiej można ocenić na przykładzie badania zgodnie ze scenariuszem3. Test ten posłużył do wyboru technologii, które najefektywniej wykorzystują przepustowość połączenia do wyświetlenia strony (100 żądań). Rysunek 2 prezentuje czasy załadowania stron a średnie wyniki zestawiono w tabeli 2.

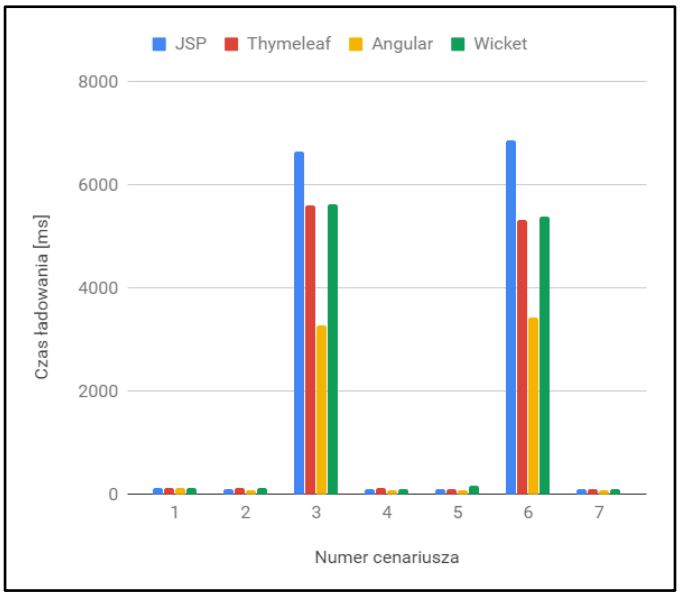


Rys. 2. Czasy wykonania 100 żądań dla S3

Tabela 2. Zagregowane dane pomiarów dla S3(zielone tło oznacza najlepszy wynik)

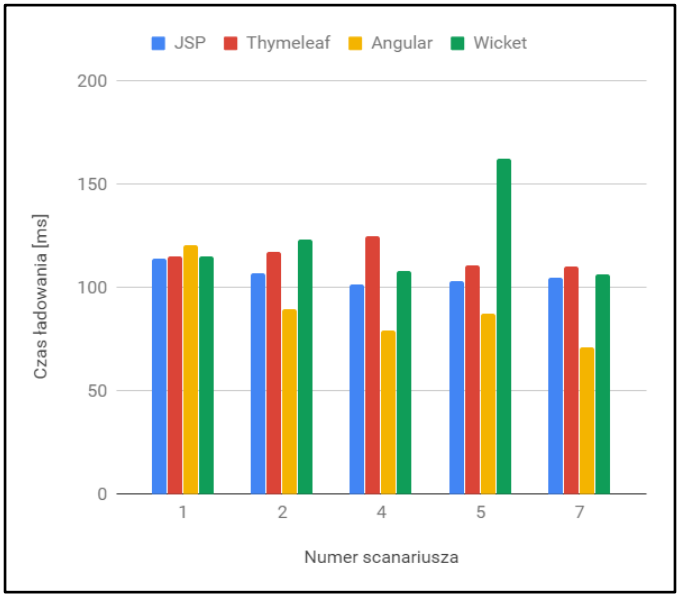
	JSP	Thymeleaf	Angular	Wicket
Minimum [ms]	3400	3899	2997	5467
Maksimum [ms]	6976	6120	5238	6827
Średnia [ms]	6467.46	5539.23	3339.18	5649.18
Mediana [ms]	6657.5	5606.5	3273	5634.5

Porównanie czasów ładowania strony dla wszystkich scenariuszy i badanych technologii jest przedstawione na rysunku3.



Rys.3. Porównanie średnich czasów wykonania żądań dla wszystkich scenariuszy

Ze względu na dużo większe wartości pomiarów czasu ładowania strony w przypadku scenariuszy z dużą liczbą elementów (3 oraz 6),wykres na rysunku 3 nie daje możliwości lepszego porównania czasów w pozostałych scenariuszach. Z tego powodu wyniki dla S1, S2, S4, S5 i S7 przedstawiono oddzielnie na rysunku 4.



Rys. 4. Średni czas wykonania żądań przez badane technologie dla scenariuszyS1, S2, S4, S5 i S7

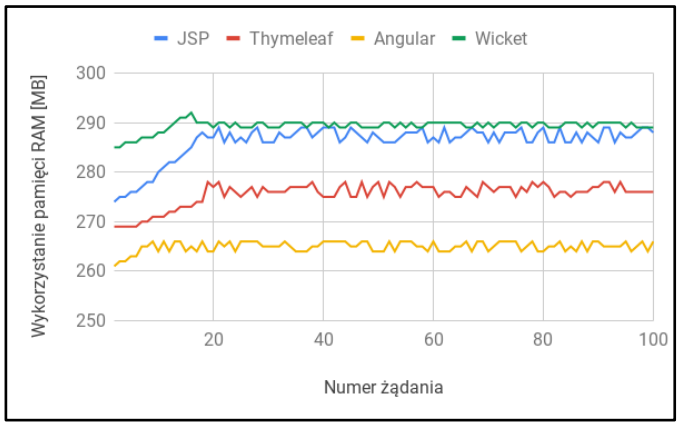
4.2. Obciążenie procesora po stronie serwera

Aplikacja Spring tworzy nowy wątek dla każdego klienta, czyli przy dużym obciążeniu w pełni są wykorzystywane zasoby procesora. Przy wszystkich opracowanych testach dla badań różnych narzędzi typu front-end używano maksymalną ilość zasobów komputera. Oznacza to, że podczas przeprowadzenia każdego badania wykorzystywano sto procent obciążenia procesora. Natomiast od strony serwera w Spring działa to w następujący sposób: system wykorzystuje wielowątkowość i każde żądanie jest obsługiwane w odrębnym wątku. Większa wydajność

konkretnej technologii front-end powoduje możliwość obsłużenia większej liczby zapytań. Ten sposób pozwolił na wskazanie wydajniejszej technologii pod względem obciążenia procesora.

4.3. Wykorzystywanie pamięci RAM

Pomiary wykorzystania pamięci RAM aplikacji działającej po stronie serwera zostały zrealizowane za pomocą Spring Actuator. Pomiary przeprowadzono podczas uruchomionego testu dla każdej badanej technologii. Porównanie wykorzystania pamięci RAM dla czterech technologii jest przedstawione na rys. 5, natomiast tabela 3 prezentuje zagregowane dane z pomiarów. Najlepsze wyniki w tym badaniu uzyskał Angular.



Rys. 5. Wykorzystanie pamięci RAM podczas wykonywania żądań do aplikacji Spring przez poszczególne technologie front-end

Tabela 3. Zagregowane dane pomiarów wykorzystywania pamięci RAM przez aplikację back-end (zielone tło oznacza najlepszy wynik)

	JSP	Thymeleaf	Angular	Wicket
Minimum [MB]	274	269	261	285
Maksimum [MB]	289	278	266	292
Średnia [MB]	286.16	275.53	264.97	289.31
Mediana [MB]	287	276	265	290

4.4. Inne cechy

Subiektywna ocena niemierzalnych cech technologii widoków została zestawiona w tabeli 4. Każde kryterium oceniono w skali od 1 do 5, gdzie większa liczba oznacza lepszą ocenę.

Tabela 4. Oceny różnych właściwości badanych technologii (zielonym kolorem są zaznaczone najwyższe oceny, czerwonym – najniższe)

	JSP	Thymeleaf	Angular	Wicket
Próg wejścia	4	5	4	2
Składnia języka	3	5	5	3
Debugowanie	3	3	5	2
Dostępne funkcjonalności	3	4	5	3
Wymaganawiedza	4	5	3	2
Wsparcie	4	4	5	1
Rozwój	3	3	5	1
Suma punktów	**24**	**29**	**32**	**14**
Język technologii	HTML + Java	HTML + znaczniki	Typescript + HTML	Java + HTML

5. Wnioski

Wobec braku oficjalnej informacji o porównaniu technologii analizowanych w niniejszej pracy, oraz braku innych publikacji na ten temat, nie ma możliwości porównania wyników własnych z wynikami z innych źródeł. Są dostępne jedynie nieoficjalne i niezatwierdzone źródła w postaci blogów oraz dyskusji w serwisach społecznościowych, wśród których często znajdują się sprzeczne wnioski, które nie mogą być traktowane jako badania naukowe.

Tezy postawione w pracy zostały potwierdzone. Szkielet aplikacji webowych Angular okazał się najbardziej efektywny, prawie we wszystkich kategoriach. Natomiast technologia Wicket nie spełniła oczekiwań i miała najgorsze wyniki. Pozostałe technologie - JSP oraz Thymeleaf pokazały nieistotną różnicę w wydajności (tabela 2, tabela 3, rys. 3, rys.4).

Thymeleaf okazał się bardzo dobrym narzędziem do tworzenia mniejszych serwisów (tabela 4), jest łatwy do obsługi na starszych urządzeniach oraz przeglądarkach, które wspierają starsze wersje języka JavaScript. Dodatkowo jest on jedną z najlepszych technologii dla migracji starszych systemów z JSP i innych technologii, bez wprowadzania dużych zmian do kodu źródłowego. W przypadku starych systemów nie spowoduje to wystąpienia dużej liczby błędów, jak na przykład w przypadku przerobienia całego systemu na architekturę REST API i konfigurowania serwerów dla dodatkowej aplikacji Angular.

Literatura

[1] Sanjay Patni: Pro RESTful APIs: Design, Build and Integrate with REST,JSON, XML and JAX-RS 1st ed. Edition. Apress 2017.

[2] Amuthan G : Spring MVC Beginner's Guide Paperback . Packt Publishing - ebooks Account 2014

[3] Craig Walls: Spring in Action - Fifth Edition. Manning Publications 2018.

[4] https://www.callicoder.com/spring-boot-actuator [13.10.2019]

[5] Hans Bergsten: JavaServer Pages - 3rd Edition. O'Reilly Media 2003.

[6] Michael Good: Thymeleaf with Spring Boot: An easy to follow guide. Amazon Digital Services LLC 2018

[7] Igor Vaynberg: Apache Wicket Cookbook. Packt Publishing 2011.

[8] Jeremy Wilken: Angular in Action 1st Edition. Manning Publications 1 edition 2018.

[9] https://siftery.com/product-comparison/angularjs-vs-thymeleaf, AngularJS vs Thymeleaf - Product Comparison [26.03.2019].

[10] https://siftery.com/product-comparison/angularjs-vs-apache-wicket, Angular vs Wicket – Product comparsion [27.03.2019].

[11] https://www.mysql.com/about, oficjalna strona MySQL [10.10.2019].

[12] https://hibernate.org/orm, oficjalna strona Hibernate ORM [10.10.2019].