Spring Framework Reliability Investigation Against Database Bridging Layer Using Java Platform

Article *in* Procedia Computer Science · December 2019

DOI: 10.1016/j.procs.2019.11.214

2 authors:

Arief Ginanjar
Langlangbuana University
13 PUBLICATIONS **14** CITATIONS
SEE PROFILE

Mokhamad Hendayun
Langlangbuana University
15 PUBLICATIONS **25** CITATIONS
SEE PROFILE

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 161 (2019) 1036–1045

Procedia Computer Science

www.elsevier.com/locate/procedia

The Fifth Information Systems International Conference 2019

Spring Framework Reliability Investigation Against Database Bridging Layer Using Java Platform

Arief Ginanjar*, Mokhamad Hendayun

University of Langlangbuana, Jl. Karapitan No. 116, Bandung, 40261, Indonesia

Abstract

There are several frameworks that can be used to make create applications easier in the Java programming environment, whether in web applications or desktop applications. If we focus more on Java web framework, there is Spring Framework that has been popular since 2004, especially with the ability of Spring Framework which can be combined with various other frameworks such as Hibernate Framework, Ibatis or namely MyBatis Framework today and several other frameworks. This research was conducted in comparing ability of data loading from a web service application built using the Java programming language with the Spring Framework, especially if combined with Database Bridging Layer such as Java Database Connection (JDBC), Hibernate Framework, MyBatis Framework, plus additional framework capabilities contained at Hibernate and MyBatis that have as cache data layer. Performance test scenario create a web service in Spring Framework then accessed by custom test script built with third party code and call it repeatedly with a certain time period.

Keywords: Spring Framework; Hibernate; Ibatis; MyBatis; JDBC; Cache Engine; Web Service. JSON.

* Corresponding author. Tel.: +62-82-121-633-200.
E-mail address: arief.ginanjar@unla.ac.id

1877-0509
10.1016/j.procs.2019.11.214

1. Introduction

1.1. Background

Spring Framework is one of the first frameworks that developed and began to be used in the implementation of making web applications using java platform in the early 2000s. Over time other frameworks have emerged that offer diverse advantages, but the popularity of Spring Framework still persists and is used until 2018. This can be seen from the following info graphics which illustrate the level of interest of java programmers to some of the most popular frameworks in the world between 2004 and 2018.

From the infographic in Fig. 1, there are some frameworks that are quite popularly used by programmers to be implemented in their work including Spring Framework, Hibernate, JSF, Struts and MyBatis, these five frameworks are the most popular among the frameworks that are used by java programmers.

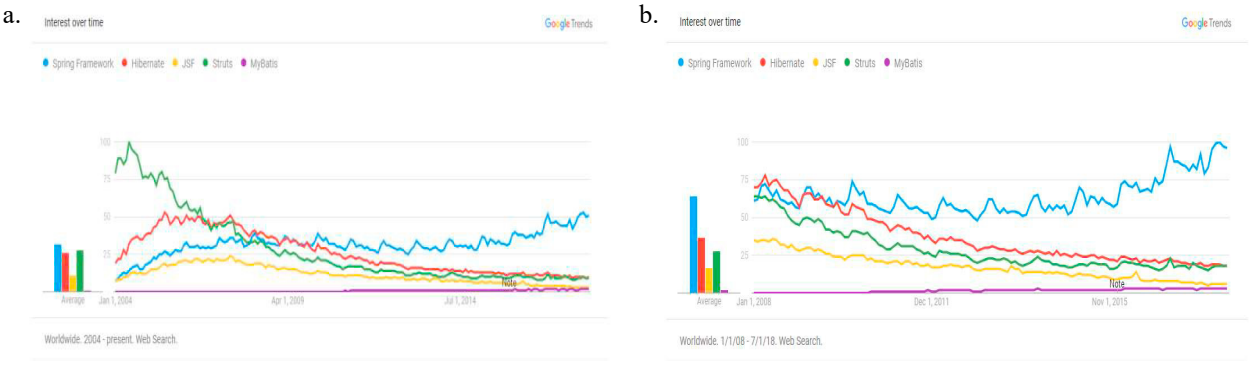


Fig. 1. Interest of several popular frameworks in the world from 2004 to 2018 (a) and 2008 to 2018; (b) (Source: Google Trend).

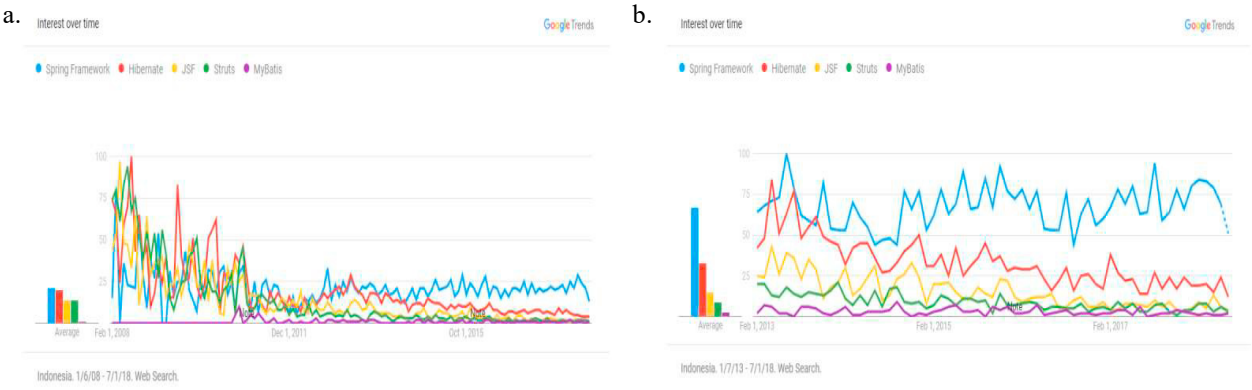


Fig. 2. Interest of several popular frameworks in Indonesia from 2008 to 2018 (a) and 2013 to 2018; (b) (Source: Google Trend).

In Fig. 2-a, there is a fluctuation in the framework of Java programmers in Indonesia between 2008 and 2010, this indicates that the framework used in the application work is still in the trial stage, but after 2011 it began to show a significant tendency for programmers to find comfort and benefit from each of the popular frameworks. Then what is seen in Fig. 2-b is the tendency of java programmers in Indonesia to five frameworks with a period of 2013 to 2018.

In implementing the use of the Spring Framework in the development of java-based applications, it is often combined with other frameworks including; Java Database Connection (JDBC), Object Relational Mapping with the Hibernate trademark, Java Query Expose Connection with the trademark Ibatis or MyBatis, and the use of a cache

engine that serves to store temporary data in addition to data stored in the database with the aim of increasing application data access speed.

1.2. Reseach goal

The research objectives to be achieved from this study are to know the reliability value of each specimen as listed in Table 1 which will be tested with a series of tests using data names of provinces, cities, districts, sub-districts and villages throughout Indonesia with 98,457 rows of data and data loading performance will be tested by loading looping on a json data url with a certain intensity.

Table 1. Framework Specimens That Will Be Tested.

Specimen	Main Framework	Bridging Framework	Cache Framework
1	Spring MVC	JDBC	-
2	Spring MVC	Hibernate	-
3	Spring MVC	Hibernate	L2 Ehcache
4	Spring MVC	MyBatis	-
5	Spring MVC	MyBatis	LRU MyBatis

With a series of tests, it is expected to be able to know the reliability of each framework when it has to face the performance test using load looping. So that it is expected to be a reference source in decision making when it comes to choosing the Spring Framework collaboration that is applied in the development of java-based applications.

1.3. Research scoping

This study only concentrates on data loading urls and does not test the insert operation, update and delete json url with the following technical specifications:

- Using the MariaDB version 10.1.19 database system placed separately in the Microsoft Windows 7 Operating System Genuine 64 bits in Oracle VM Virtual Box version 5.2.6.
- Using network connection between local host - guest virtualbox.
- Using Microsoft Windows 7 Genuine 64-bit operating system for host environment.
- Using Netbean 8.2 as Integrated Development Editor.
- Using Spring Framework version 4.0.1 as MVC Framework.
- Using Spring-json taglib as library for json output.
- Using application container Apache Tomcat version 9.0.12 as application deployment.
- Using Java Virtual Machine version 1.8.0.162 64 bit with setting core JVM core default setting without any adjustment as platform environment.

2. Literature review

2.1. Research methodology

The methodology used in the preparation of this research report is to use quantitative research methods and prototypes as well as the process of testing the framework's performance capabilities with repetition of processes and systems approaches emphasizing elements and components [1,2, 3], while the process sequence is carried out as follows:

- Literature Study, Studying library resources that can be used as references. Library sources can be books, papers or web pages that discuss Spring Framework, Hibernate Framework, Ibatis Framework and Cache Engine.

- Analysis, Describes how to do an analysis of the architecture and programming techniques for each framework and how the technique combines each of these frameworks.
- Software Design, Doing software design combined between frameworks that will be built based on the results obtained from the analysis. The design must meet the testing scenario that will be carried out.
- Software Implementation, Implementing software that will be developed based on the results obtained from the design. This implementation will produce software products that contain a combination of frameworks as specified in the specimen to be tested.
- Testing and Evaluation, Conducting testing of software products that have been built and then performing a performance evaluation of each test scenario conducted.
- Iterative Software Design and Implementation Stages as well as Testing and Evaluation, Conducting testing of software specimens that will be examined when the business process has not fulfilled the wishes of the researcher then do the design and re-implementation of specimen software.
- Application Performance Testing, Conducting the testing process of the application's ability to pressure by url loop loading which is done repeatedly against specimen frameworks that have been built into applications using the url looping script method with java programming scripts built by researchers.

2.2. Evolutionary prototyping model

With the process stages described in the implementation section of the study that refer to the evolutionary stages of prototyping models, the process can be illustrated as shown in Fig. 3. The evolutionary process of prototyping consists of four main processes, input, prototyping process, and output, but during the process must also be limited by conditions; each function built must meet the appropriate requirements as specified in the system requirements, with the competence of the people involved in the process of prototyping methodology meets the minimum system requirements [4].

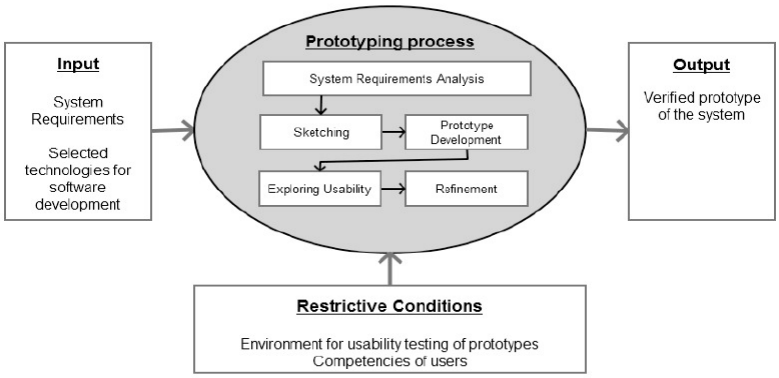


Fig. 3. Evolutionary Prototyping Model [4].

2.3. Specimen architecture

With an approach using the principles of Object Oriented Programming as well as the Model, View and Controller MVC approach, abstraction allows programmers to develop complex thinking without having to focus on detailed components, while encapsulation allows us to focus on software capabilities without having to think in detail about the complexity of the process happen. [5, 6].

By combining the Spring Framework, Hibernate Framework and MyBatis Framework architectures into a system that overlaps the shortcomings of each framework, it is expected that the tests that can be carried out and can be found to be optimized in combining these frameworks. [7, 8, 9].

a. b.

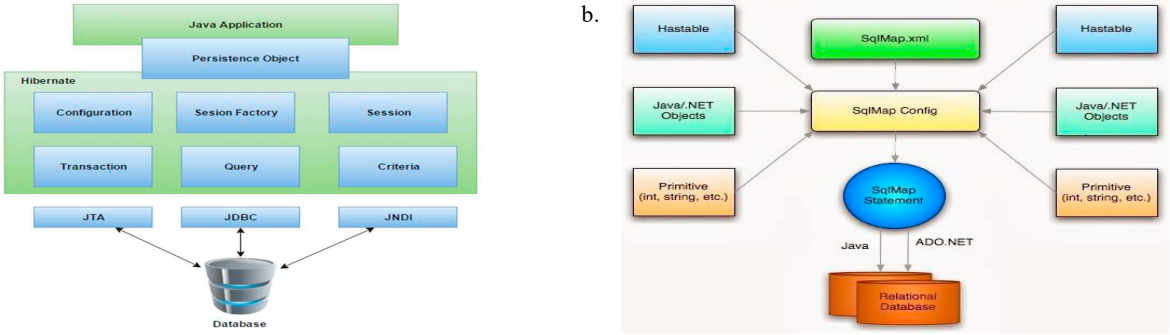


Fig 4. Hibernate Framework Architecture (a) [9]; and MyBatis Framework Architecture (b) [8].

3. Research methodology

3.1. Research background

When develop a software using evolutionary prototyping, the steps that are carried out start from analysis, design and implementation, testing and evaluation repeatedly with restrictive conditions to produce the desired output. The aim of finding the best combination of frameworks to produce web service generators in the java platform.

3.2. Specimen configuration

In testing specimens that have been carried out there are technical aspects that have been applied, namely; Tests carried out involve several system layers, the database layer, application container layer, logic programming layer and test script layer. Then in each layer there are several configurations that are used so that each system in the layer can interact with other layers.

Table 2. Configuration Implementation at Each *Layer*.

No	Layer	Technology	Configuration
1	Database	MySQL	Default
2	Application Container	Apache Tomcat	Default
3	Logic Programming	Spring Framework Combination	Specifically each specimen
4	Script Pengujian	Java Programming	Specifically each specimen

Then the test flow of each specimen stored on the virtual server installed in the guest OS, then Apache Tomcat is installed in the host OS, followed by the test script run using Netbean IDE version 8.2 with illustrations shown in Fig. 5. Each framework specimen seen in Table 1 requires also several other libraries as support for the main framework in research specimens, with library details can be seen in Table 3.

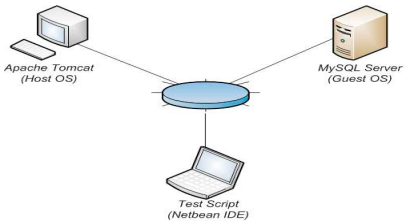


Fig 5. Spring framework data flow architecture tests performance for several database bridging layers.

As seen in the specimen column 1 which is a test specimen for the Spring Framework with version 4.0.1 combined with Java Database Connection (JDBC), it requires another library, the JSP Standard Tag Library (JSTL) which has a function to translate data from the java class environment to the environment java web, then MySQL-JDBC is a Java Database Connection (JDBC) specifically for database servers with MySQL trademarks, then also assisted with JSON Taglib, a converter tool that functions to convert data from the JSTL environment to Javascript Object Notation.

Table 3. Additional Library for Each Specimen.

Library	Specimen				
	1	2	3	4	5
JSTL	√	√	√	√	√
MySQL JDBC	√	√	√	√	√
JSON Taglib	√	√	√	√	√
Persistence JPA 2.1	×	√	√	×	×
AOP Alliance	×	√	√	×	×
Ehcache	×	×	√	×	√

Whereas Persistence JPA 2.1 is the main support tool used as a bridge between the Spring Framework and Hibernate Framework so that data exchange between the two frameworks is effective, then AOP Alliance becomes a complementary function of Persistence JPA 2.1, when there are several functions that cannot be facilitated by Persistence JPA 2.1 it will be assisted by the AOP Alliance to cover interoperability needs. Ehcache is a technology based on hardware or software that is used to be used as temporary data storage that is used to help the level of performance of an application or system.

The application of cache technology is already quite widely used by application developers to increase the speed of access to application data views, each framework has its own architecture in implementing the cache to the data it manages, it is quite interesting to be the next research material.

3.3. Data gathering

Data collection carried out by the researcher was in the form of running application specimens built from the composition of the framework as shown in Table 1 and installed in the application container environment with the product name Apache Tomcat version 9.0.12 then run using test script in the Netbean 8.2 environment as seen Fig. 6.

```
public static void main(String[] args) throws IOException, JSONException {
    // TODO code application logic here
    long secondStart = System.currentTimeMillis();
    JSONArray jsonArray = readJsonArrayFromUrl("http://localhost:8080/Research-Spring-Jdbc/json.htm");
    for (int i = 0; i < jsonArray.length(); i++) {
        //get the JSON Object
        JSONObject obj = jsonArray.getJSONObject(i);
        Integer id = obj.getInt("id");
        String nama = obj.getString("nama");
        Integer anak = obj.getInt("anak");
        System.out.println("id = "+id+", nama = "+nama+", anak = "+anak);
    }
    long secondFinish = System.currentTimeMillis();
    long elapseTime = secondFinish - secondStart;
    System.out.println(" second "+elapseTime);
}
```

Fig 6. Command line tests the spring framework performance for several database bridging layers using java programming.

4. Result

4.1. Research result

During the performance testing of the Spring Framework specimens combined with the Database Bridging Framework it was found that there was a positive tendency that occurred between the combination of the MyBatis Framework and L2 Cache, but there was a negative tendency towards the combination of Hibernate Framework and L2 Cache.

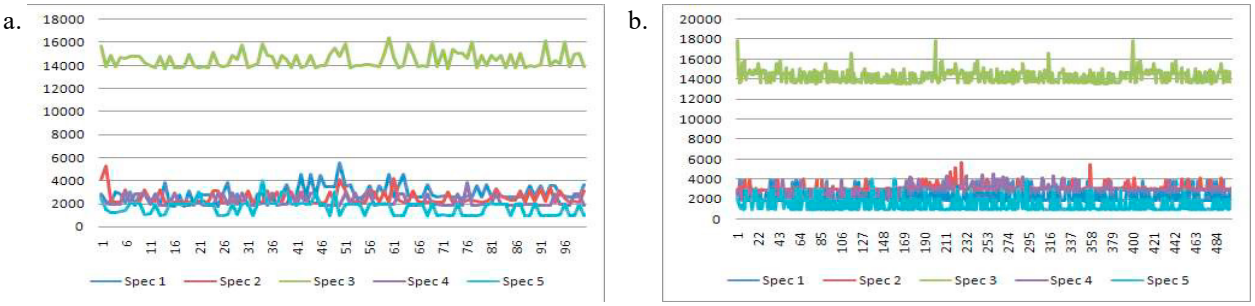


Fig 7. Scenario one test results for 100 hits (a); and 500 hits (b).

From the tests that have been carried out based on the predetermined test scenario, it is found that the three specimens that contain the Spring Framework, Hibernate Framework and 2nd Level Ehcache cannot be executed when hits are above 1000 for scenario one and scenario two, then with the same specimen not can be run to hit above 500 for scenarios three, four, five and six.

With the above conditions, the writer is enough to focus on the hit 100 data and hit 500 with the aim to give the balance of the results of the observation data for all specimens and scenarios. In the next discussion the representation of the data displayed has been converted into a line diagram as shown in Fig. 7 to Fig. 12.

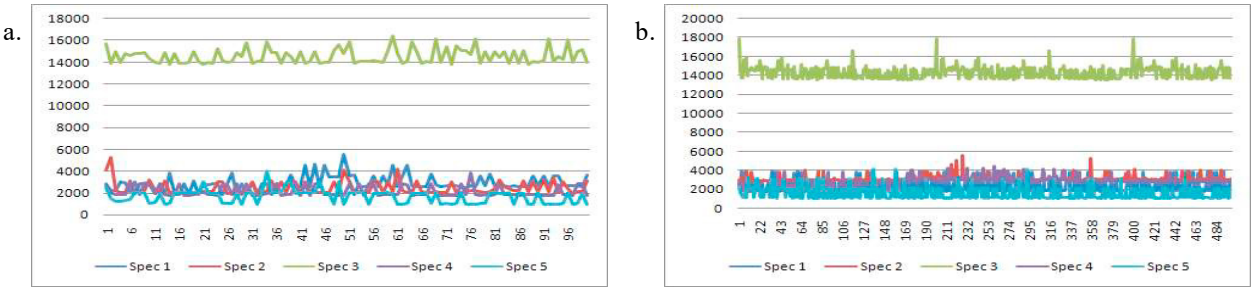


Fig 8. Scenario two test results for 100 hits (a); and 500 hits (b).

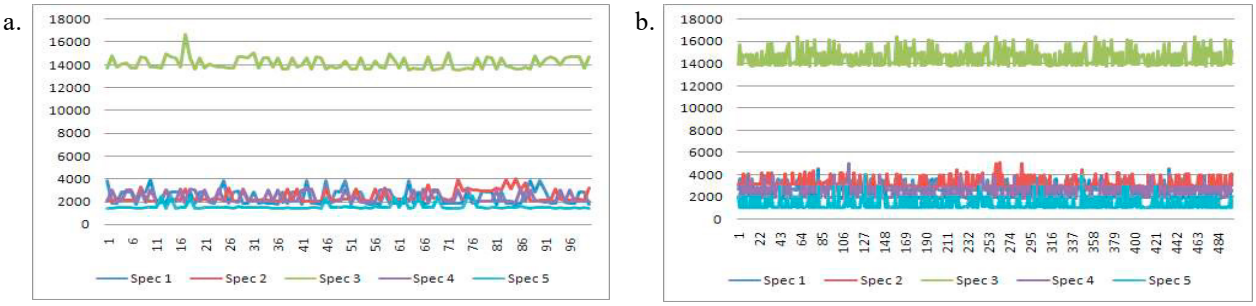


Fig 9. Scenario three test results for 100 hits (a); and 500 hits (b).

a.

b.

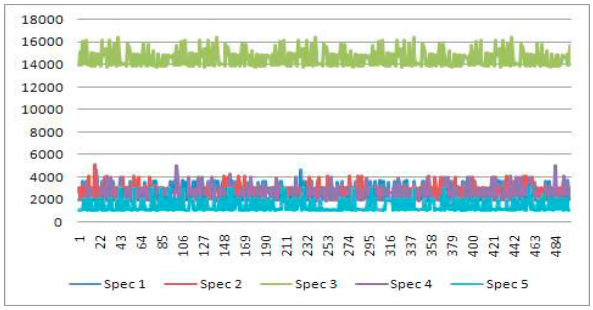
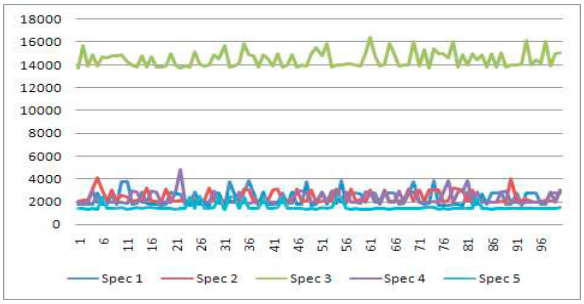


Fig 10. Scenario four test results for 100 hits (a); and 500 hits (b).

a.



b.

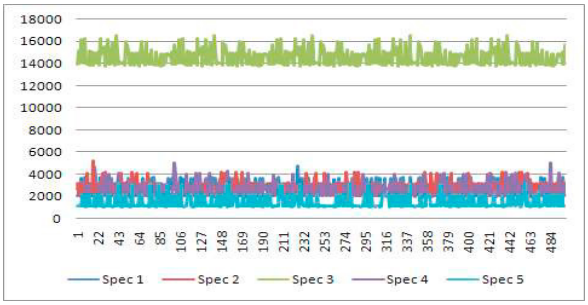
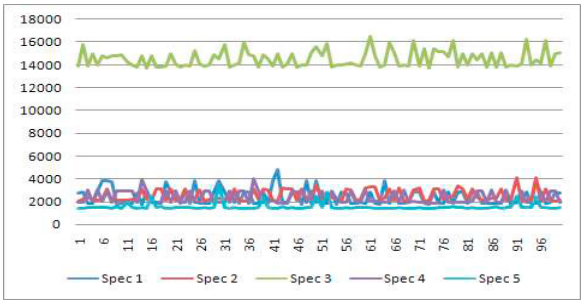


Fig 11. Scenario five test results for 100 hits (a); and 500 hits (b).

a.



b.

Fig 12. Scenario six test results for 100 hits (a); and 500 hits (b).

From the data listed in the figure which is represented by a line diagram as seen above, the researcher will take the main data, namely; average load speed for each scenario vs. each specimen, lowest load speed for each scenario vs each specimen, highest load speed for each scenario vs each specimen, load speed equal to the most frequently occurring in each scenario vs each specimen, and load speed value middle in each scenario vs each specimen. All data are presented in Table 4, Table 5 and Table 6.

Table 4. Access Speed Results of Scenario One and Two in Each Specimen.

Specimen	Load Speed (milisecond)				Specimen	Load Speed (milisecond)			
	Average	*Slow*	*Fast*	*Modus*		*Average*	*Slow*	*Fast*	*Modus*
1	2,851.27	5,526.00	1,750.00	2,603.00	1	2,887.16	5,560.00	1,760.00	2,817.00

Specimen		Load Speed (milisecond)				Specimen		Load Speed (milisecond)			
		Average	*Slow*	*Fast*	*Modus*			*Average*	*Slow*	*Fast*	*Modus*
Scenario 1 Hit 100	2	2,483.76	5,312.00	2,009.00	2,094.00	*Scenario 2 Hit 100*	2	2,447.87	5,292.00	1,960.00	2,083.00
	3	14,501.3	16,428.0	13,729.0	14,928.0		3	14,537.2	16,438.0	13,774.0	14,814.0
	4	2,189.44	3,869.00	1,827.00	1,871.00		4	2,153.55	3,846.00	1,785.00	1,810.00
	5	1,667.76	3,995.00	954.00	1,972.00		5	1,703.65	4,015.00	984.00	2,013.00
Scenario 1 Hit 500	1	2,308.89	3,907.00	1,711.00	1,789.00	*Scenario 2 Hit 500*	1	2,344.78	3,957.00	1,748.00	1,834.00
	2	3,098.85	5,638.00	2,183.00	3,041.00		2	3,062.96	5,603.00	2,133.00	2,938.00
	3	14,291.1	17,854.0	13,516.0	13,569.0		3	14,326.9	17,889.0	13,536.0	13,745.0
	4	2,823.50	4,526.00	1,831.00	2,930.00		4	2,787.61	4,486.00	1,774.00	2,871.00
	5	1,518.25	4,044.00	987.00	1,021.00		5	1,554.14	4,094.00	1,006.00	1,058.00

Table 5. Access Speed Results of Scenario Three and Four in Each Specimen.

Specimen		Load Speed (milisecond)				Specimen		Load Speed (milisecond)			
		Average	*Slow*	*Fast*	*Modus*			*Average*	*Slow*	*Fast*	*Modus*
Scenario 3 Hit 100	1	2,400.45	3,920.00	1,769.00	2,844.00	*Scenario 4 Hit 100*	1	2,328.50	3,853.00	1,735.00	1,796.00
	2	2,466.06	4,029.00	2,035.00	2,164.00		2	2,485.16	4,283.00	1,985.00	2,110.00
	3	14,141.41	16,692.0	13,536.0	13,817.0		3	14,417.9	17,854.0	13,564.0	14,656.0
	4	2,328.62	3,085.00	1,928.00	2,007.00		4	2,295.79	3,969.00	1,898.00	2,010.00
	5	1,587.41	2,522.00	1,421.00	1,468.00		5	1,566.00	2,873.00	1,398.00	1,443.00
Scenario 3 Hit 500	1	2,812.76	4,601.00	2,495.00	2,613.00	*Scenario 4 Hit 500*	1	2,816.69	4,668.00	2,492.00	2,657.00
	2	3,198.15	5,070.00	2,766.00	3,136.00		2	2,608.31	5,132.00	1,998.00	3,080.00
	3	14,501.30	16,428.0	13,729.0	14,928.0		3	14,537.2	16,448.0	13,764.0	16,154.0
	4	2,369.24	5,043.00	1,864.00	1,916.00		4	2,414.12	4,999.00	1,882.00	1,974.00
	5	1,370.43	4,025.00	987.00	1,019.00		5	1,408.47	3,100.00	1,004.00	1,048.00

Table 6. Access Speed Results of Scenario Five and Six in Each Specimen.

Specimen		Load Speed (milisecond)				Specimen		Load Speed (milisecond)			
		Average	*Slow*	*Fast*	*Modus*			*Average*	*Slow*	*Fast*	*Modus*
Scenario 5 Hit 100	1	2,315.25	3,830.00	1,694.00	1,770.00	*Scenario 6 Hit 100*	1	2,432.19	4,868.00	1,752.00	1,844.00
	2	2,387.60	4,074.00	1,974.00	2,054.00		2	2,543.92	4,152.00	2,051.00	2,160.00
	3	14,463.10	16,388.0	13,686.0	14,764.0		3	14,539.2	16,470.0	13,751.0	13,869.0
	4	2,273.23	4,868.00	1,837.00	1,972.00		4	2,397.68	4,053.00	1,903.00	1,959.00
	5	1,537.26	2,501.00	1,336.00	1,399.00		5	1,586.87	3,489.00	1,438.00	1,470.00
Scenario 5 Hit 500	1	2,852.58	4,746.00	2,517.00	2,674.00	*Scenario 6 Hit 500*	1	2,819.11	4,686.00	2,486.00	2,613.00
	2	2,644.20	5,177.00	2,023.00	2,132.00		2	2,677.67	5,212.00	2,037.00	2,135.00
	3	14,573.08	16,498.0	13,794.0	14,106.0		3	14,539.6	16,488.0	13,739.0	13,981.0
	4	2,450.01	5,031.00	1,912.00	1,980.00		4	2,483.48	5,041.00	1,932.00	2,040.00
	5	1,444.36	3,139.00	1,014.00	1,088.00		5	1,410.89	3,113.00	963.00	1,058.00

4.2. Discussion result

The speed of data access to each specimen is compared with each scenario based on data from Table 4, Table 5 and Table 6 notice that:

- Specimens four and five from each test scenario have access speeds which are quite significant both in the average speed in each looping access and the highest speed that can be achieved in some looping access.

- Specimens one and two in some test scenarios have a high enough speed but are not high enough compared to specimens four and five from each test scenario, while specimens one and two are high enough for all test scenarios with 100.
- Specimens three for each test scenario have a low speed tendency in all test scenarios.
- For the highest average speed can be seen in the scenario of three hits 500 in the five specimens with a speed of 1,370.45 milisecond.
- Whereas for the highest speed of access can be seen in the scenario of one hit 100 in the fifth specimen with a speed of 954 milisecond.
- For the lowest average speed of access can be seen in scenario five hits 500 on specimen 3 at a speed of 14,573.08 milisecond.
- Whereas for the lowest access speed with comparative discussion data number six, scenario five hits 500 on specimen 3 at a speed of 13,794 milisecond.

From the interim conclusions derived from Table 4, Table 5 and Table 6, researchers can draw conclusions that the 5th specimens at top speed, followed by 4th specimens, then 1st and 2nd specimens then specimen 3rd is in the last position in the access speed.

5. Conclusion

With research on investigating the performance of the Spring Framework against several Bridging Frameworks, focusing only on data loading urls that have been done, there are several results that can be known, namely;

- The merge between Spring Framework 4.0.1 and MyBatis Framework 3.4.2 produces very positive performance, then when added to the Cache Engine performance increases.
- The mix of Spring Framework 4.0.1 with Hibernate Framework 4.3 produces positive performance, but changes direction to negative when added to the 2nd Level Cache.
- Whereas the Spring Framework 4.0.1 mix with the Java Database Connection (JDBC) tends to be moderate with the results of the testing being in the middle, not negative and not positive.
- Therefore, researchers tend to choose the combination of the Spring Framework and MyBatis Framework as a high-performance web service generator, specially with cache feature addition.

Acknowledgements

This research was partially funded by Department of Informatics, Faculty of Engineering, University of Langlangbuana.

References

[1] Begin, Clinton. (2007) *Ibatis in Action*, Manning Publication, New York, USA.
[2] Jogiyanto, H.M. (2008) *Analisis dan Perancangan Sistem: Pendekatan Terstruktur Teori dan Praktek Aplikasi Bisnis* [Title in English: *System Analysis and Design: Structured Approach including Theory and its Application in Busines*], Andi Offset., Yogyakarta., Indonesia.
[3] Johnson, Rod. (2005) *Professional Java Development with the Spring Framework*, Wiley Publishing., Indianapolis, USA.
[4] Kalelkar, Medha., Churi, Prathamesh., Kalelkar, Deepa. (2014) Implementation of Model-View-Controller Architecture Pattern for Business Intelligence Architecture., International Journal of Computer Application., New York, USA.
[5] Kendal, Simon. (2009) *Object Oriented Programming using Java*., Ventus Publishing Aps, Frederiksberg, Denmark.
[6] Sugiyono. (2012) *Metode Penelitian Kuantitatif Kualitatif dan R&D* [Title in English: *Qualitative and Quantitative Research Method and R&D*], Alfabeta, Bandung. Indonesia.
[7] Zimanyi, Esteban. (2014) *Introduction to Java Hibernate*, Universite libre de Bruxelles, Bruxelles, Belgia.
[8] Roger, S. Pressman. (2012) Rekayasa Perangkat Lunak [Title in English: *Software Engineering*], 7th Edition : Book 1, Andi Offset, Yogyakarta., Indonesia.
[9] Nacheva, Radka. (2017) *Prototyping Approach in User Interface Development, Conference on Innovative Teaching Methods*, University of Economics Varna, Bulgaria.