Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 198 (2022) 282–286

Procedia Computer Science

www.elsevier.com/locate/procedia

The 1st International Workshop on Design and Use of Digital Services to Empower Healthcare and Wellbeing (DUDE 2021)
November 1-4, 2021, Leuven, Belgium

Making Java EE Cool Again: Building MongoDB-Based Web Services Using JPA and EJB

Fabian Kaimer[a], Philipp Brune*[a]

[a]*Neu-Ulm University of Applied Sciences, Wileystraße 1, 89231 Neu-Ulm, Germany*

Abstract

Despite its dominating role for developing large-scale enterprise applications, Java Enterprise Edition (EE) has always been seen critically by many developers, in particular from the web and open source communities. In recent years, the discussion about its decline and replacement e.g. by scripting languages such as Python or JavaScript has therefore gained momentum. On the other hand, Java EE has evolved towards being more flexible, lightweight and simpler to use and offers unique features still not available out-of-the-box on other platforms, such as support for distributed transactions. Therefore, in this paper an approach is presented, how a Java EE-based application supporting elderly people could be designed to make use of a modern NoSQL database (namely MongoDB), while preserving its unique features. The presented approach demonstrates not only the feasibility of such an architecture, but also supports the claim that Java EE still is a highly relevant and powerful platform for building modern applications.

Keywords: Web Services; Java EE; MongoDB; Java Persistence API; Enterprise Java Beans

1. Introduction

In recent years, amongst software developers a discussion came up regarding the future of the Java programming language including its libraries and associated platforms. Some authors suggest that the Java ecosystem in total will lose its importance [18, 20], and may eventually be replaced by more modern languages and technologies like Python, JavaScript or Node.js [8], especially in the domain of web-based systems.

* Corresponding author. Tel.: +49-731-9762-1503; fax: +49-731-9762-1599.
E-mail address: Philipp.Brune@hnu.de

1877-0509
10.1016/j.procs.2021.12.241

Surveys show, that in 2018 JavaScript is the most popular language amongst ”stackoverflow.com” users [22] and the most frequently used programming language by software developers worldwide [11].

”Java Platform, Enterprise Edition” (Java EE), a Java software architecture specification for the creation of large-scale enterprise applications, has even been prognosed to become obsolete in the future and will lose its importance in the upcoming cloud-native application context [23].

Indeed, a shift in the field of software development, concerning utilized programming languages, technologies and platforms, in the area of web-based systems, can be observed (e.g. the increased utilization of server-side JavaScript or the increasing use of NoSQL databases over relational databases). Additionally, Oracle’s transfer of the Java EE platform development from a company-backed process to an open-source community-driven development process under the new name ”Jakarta EE” has caused uncertainty both amongst developers and companies.

Nevertheless, neither the increasing use of these new technologies nor the organizational changes that the specification is currently undergoing do mark the end for a proven platform like Java EE. In fact, it opens new and interesting possibilities of interoperability, if both - old and new technologies - are combined to generate synergetic effects like increased speed and performance or enhanced application security.

Therefore, this paper demonstrates the feasibility of such an approach by example of a server back-end for a mobile app consisting of a Java EE application in combination with a MongoDB database.

The rest of this paper is organized as follows: In section 2, the related work is analyzed in more detail, while the research methodolgy used is outlined in section 3. Section 4 briefly introduces Java EE. In section 5 the proposed software architecture is described, and section 6 oulines its planned evaluation. We conclude with summary of our findings.

2. Related Work

In recent years, the Java EE specification has been a topic in science as well as in practice. In terms of practical usage, multiple commercial [9, 19] and non-commercial frameworks [14, 25] have been implemented, based on the different versions of Java EE.

Early scientific approaches dealing with Java EE mainly feature different implementation approaches for the specification [6, 12, 21] or consider best practices for application development on the platform [17].

More recent publications either present concrete projects based upon a Java EE framework [5, 4] or concern its features, like security [7, 13] or integration with third-party technologies [2].

Although there exist scientific approaches dealing with Java EE applications in a context of a rapidly changing technological environment, none of them have yet dealt with the combination of Java EE and MongoDB. Additionally, none of the authors questions the utilization of a Java EE implementation in an environment of newer technologies under the aspect of the current changes that the specification is undergoing.

Therefore, in this paper the question is addressed: “How can a software architecture be designed to enable the connection of a Java EE application with a MongoDB database as an example of demonstrating Java EE’s relevancy in the future?”

3. Research Methodology

The scientific methodology of the paper is following the ”Design Science Research” approach [24]. First, a brief analysis of the topic’s related literature to identify the research gap has been carried out, leading to the research question. Then, the Java EE specification and its significance are examined more closely. An exemplary application architecture for the connection of a Java EE application with a MongoDB database is being proposed and implemented to demonstrate that Java EE can still be relevant amongst rapidly emerging new technology. Finally, the future evaluation of the definitive architecture through a load testing, following the approach of Brune [3], is being outlined.

4. Java EE

”Java Platform, Enterprise Edition” (Java EE), which is currently available at version 8, represents a standardized specification for the development of large-scale, transaction-based enterprise-level (web-)applications. The specifi-

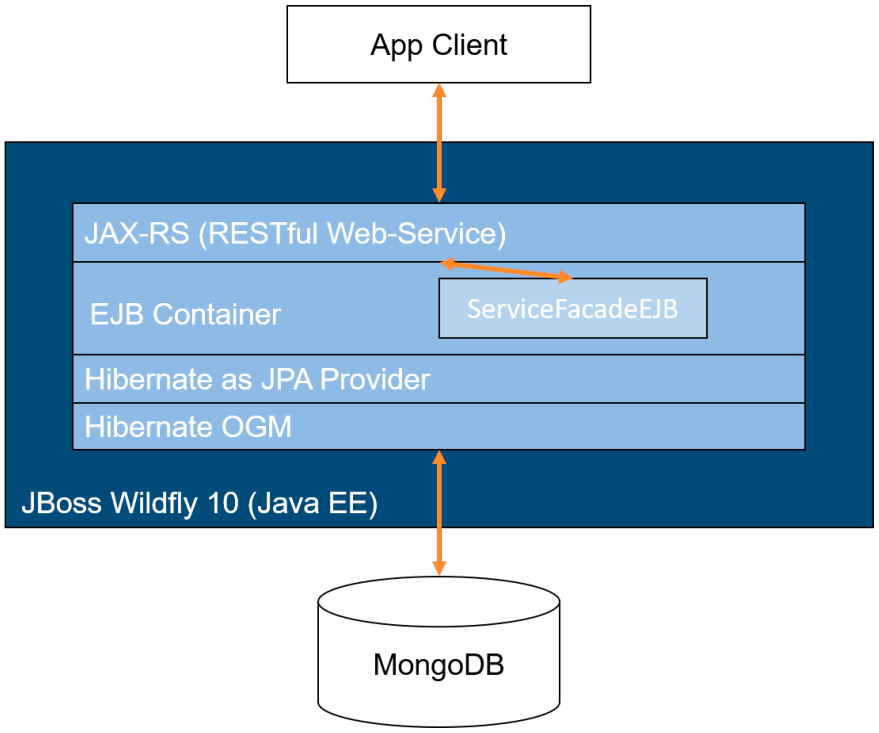

Fig. 1. Hibernate as JPA provider and Hibernate OGM server as an interface to connect the Java EE application to the MongoDB database

cation defines several modular services, their interfaces and the way they communicate which qualifies for highly flexible application engineering.

Thereby, all the components offer a high level of interoperability and scalability within the context of their whole system. Every Java EE application is divided into several logical units, called ”containers” (e.g. EJB-Container, Web-Container, JMS-Provider), which need to be existent in every framework implementation of the specification [14, 16].

Nowadays, multiple commercial (e.g. IBM WebSphere [10], JBoss EAP [19]) and open source (e.g. GlassFish [15], JBoss WildFly [25], Apache Geronimo [1]) implementations of the Java EE specification exist. The latest - and probably last - Java EE specification (version 8) has been released in 2017, with its first implementation being OpenLiberty by IBM [9].

5. Software Architecture

The proposed application architecture represents the server back-end for a real-world mobile app. It is built upon ”JBoss Wildfly 10”, an open-source application server, that implements the Java EE standard and is operated on a Java Virtual Machine (JVM). A ”Java API for RESTful Web Services” (JAX-RS) is operated as a web interface, enabling the application to provide the RESTful web service. An Enterprise JavaBeans-Container (EJB) constitutes the actual application core. It contains the ”ServiceFacadeEJB” Java class that processes the app client’s server calls. To enable the application to connect to the MongoDB, the open-source framework ”Hibernate” operates as the application’s persistence layer. It is implemented as Java Persistence API (JPA) provider towards the application and connected to Hibernate Object/Grid Mapper (OGM) that provides the MongoDB interface (see Fig. 1). As seen in Listing 1, the database query for the NoSQL storage only differs slightly from a traditional OGM query (Listing 2), while providing the same functionality as the traditional approach.

```
@GET
@Path("getChangedObjects/{userId}/{userToken}/{deviceId}/{ownerUserId}")
@Produces({ "application/json" })
        public List<DataObject> getChangedObjects(
                @PathParam("userId") String userId,
                @PathParam("userToken") String userToken,
...
        Query q = em.createNativeQuery("{$and:[{$or:[{ownerUserId:\""
                + ownerUserId
                + "\"},{ownerUserId:\"global\"}]},{$or:[{state:1},{state:2}]},"
                + "{updatedDevices:{$nin:[\"" + deviceId
                + "\"]}}]}", DataObject.class);
        List<DataObject> res = q.getResultList();
...
        return res;
        }
}
```

Listing 1. A "GET"-function query inside the ServiceFacadeEJB that uses the Hibernate OGM API

```
@GET
@Path("createNutzer/{userId}/{pwHash}/{pubKey}/{code}/{deviceId}/{isArzt}")
@Produces({ "application/json" })
        public Nutzer createNutzer(
                @PathParam("userId") String userId,
                @PathParam("pwHash") String pwHash,
                @PathParam("pubKey") String pubKey,
...
        TypedQuery<Nutzer> q2 = em.createQuery(
                        "SELECT u FROM Nutzer u WHERE u.userId =:uId AND"
                        + "(u.state = 0 OR u.state = 1)",
                        Nutzer.class);
        q2.setParameter("uId", userId);
        List<Nutzer> nutzerListe = q2.getResultList();
...
        return Nutzer;
        }
}
```

Listing 2. A "GET"-function inside the ServiceFacadeEJB that uses the Hibernate ORM as Java Persistence API

6. Planned Evaluation

The feasibility of the proposed software architecture approach for the connection of Java EE applications with MongoDB databases has been demonstrated through its implementation and usage as an application back-end for a real-world mobile app for Android and iOS and will furtherly be proven by its upcoming load testing evaluation.

The evaluation will simulate an increasing number of app users with an increasing number of connections between them, depending on factors like daily activity or location and their interactions amongst each other resulting in app back-end calls, following the approach of Brune [3].

7. Conclusion

In conclusion, in this paper examplified architecture for the development of web services has been presented, based on Java EE and MongoDB using EJBs, Hibernate ORM as JPA provider and Hibernate OGM as the database access layer.

The successful implementation of the architecture shows its feasibility and demonstrates that the Java EE platform can still be relevant in the rapidly changing environment of more modern software technologies and platforms like Node.js or MongoDB.

References

[1] Apache Foundation, . Apache geronimo. URL: http://geronimo.apache.org/.

[2] Brune, P., 18.09.2018 - 20.09.2018. A hybrid approach to re-host and mix transactional cobol and java code in java ee web applications using open source software, in: Proceedings of the 14th International Conference on Web Information Systems and Technologies, SCITEPRESS - Science and Technology Publications. pp. 239–246. doi:10.5220/0006943402390246.

[3] Brune, P., 25.04.2017 - 27.04.2017. Simulating user interactions: A model and tool for semi-realistic load testing of social app backend web services, in: Proceedings of the 13th International Conference on Web Information Systems and Technologies, SCITEPRESS - Science and Technology Publications. pp. 235–242. doi:10.5220/0006248202350242.

[4] Brune, P., Leiser, M., Janke, E., 2014. Towards an easy-to-use web application server and cloud paas for web development education, in: Proceedings, 16th IEEE International Conference on High Performance Computing and Communications, HPCC 2014 ;, Conference Publishing Services, IEEE Computer Society, Los Alamitos, California and Washington and Tokyo. pp. 1113–1116. doi:10.1109/HPCC.2014.187.

[5] Chen, S., Song, B., 24.04.2015 - 26.04.2015. Research on e-learning system based on java ee architecture, in: Proceedings of the 2015 International Conference on Education, Management, Information and Medicine, Atlantis Press, Paris, France. doi:10.2991/emim-15.2015.27.

[6] Desertot, M., Donsez, D., Lalanda, P., 2006. A dynamic service-oriented implementation for java ee servers, in: IEEE International Conference on Services Computing, 2006, IEEE Computer Society, Los Alamitos, Calif. [u.a.]. pp. 159–166. doi:10.1109/SCC.2006.4.

[7] Guamán, D., Guamán, F., Jaramillo, D., Correa, R., 2016. Implementation of techniques, standards and safety recommendations to prevent xss and sql injection attacks in java ee restful applications, in: Rocha, Á., Correia, A.M., Adeli, H., Reis, L.P., Mendonça Teixeira, M. (Eds.), New Advances in Information Systems and Technologies. Springer International Publishing, Cham. volume 444 of *Advances in Intelligent Systems and Computing*, pp. 691–706.

[8] Harishankar Karunanidhi, 2017. Bye bye java: It's time for indian it institutions to let go of outdated coding languages: Move java to javascript. URL: https://qz.com/india/1084770/bye-bye-java-its-time-for-indian-it-institutions-to-let-go-of-outdated-coding-languages/.

[9] IBM, a. Openliberty. URL: https://openliberty.io/.

[10] IBM, b. Websphere application server (distributed and ibm i operating systems): Version 8.5.5 documentation. URL: https://www.ibm.com/us-en/marketplace/java-ee-runtime.

[11] JetBrains, 2018. Programming languages used by software developers worldwide as of 2018, by deployment type. URL: https://www.statista.com/statistics/869092/worldwide-software-developer-survey-languages-used/.

[12] Liu, S., Chen, P., 2009. Developing java ee applications based on utilizing design patterns, in: Qiu, R. (Ed.), WASE International Conference on Information Engineering, 2009, IEEE, Piscataway, NJ. pp. 398–401. doi:10.1109/ICIE.2009.151.

[13] Martínez, S., Cosentino, V., Cabot, J., 2016. Model-based analysis of java ee web security configurations, in: Proceedings of the 8th International Workshop on Modeling in Software Engineering - MiSE '16, ACM Press, New York, New York, USA. pp. 55–61. doi:10.1145/2896982.2896986.

[14] Oracle, a. URL: https://www.oracle.com/technetwork/java/javaee/overview/index.html.

[15] Oracle, b. URL: https://www.oracle.com/technetwork/middleware/glassfish/overview/index.html.

[16] Oracle, c. Jsr-000366 javatm platform, enterprise edition 8 (java ee 8) specification: Final release. URL: https://jcp.org/aboutJava/communityprocess/final/jsr366/index.html.

[17] Prajapati, H.B., Dabhi, V.K., 2009. High quality web-application development on java ee platform, in: IACC 2009, IEEE, [Piscataway, N.J.]. pp. 1664–1669. doi:10.1109/IADCC.2009.4809267.

[18] Rajasekar Elango, 2018. Java is too old, what should you learn in 2018? URL: https://hackernoon.com/java-is-too-old-what-should-you-learn-in-2018-12cd0151b2d1.

[19] Red Hat, . Red hat jboss enterprise application platform. URL: https://developers.redhat.com/products/eap/overview/.

[20] Rob Muhlestein, 2018. Java is dying: Stop using and teaching it. URL: https://medium.com/@robmuh/java-is-dying-78680d34718e.

[21] Song, B., Zhang, Y., Zhou, C.S., 2010. Implementation on network teaching system based on java ee architecture, in: Second International Conference on Information Technology and Computer Science (ITCS), 2010, IEEE, Piscataway, NJ. pp. 227–231. doi:10.1109/ITCS.2010.62.

[22] Stackoverflow.com, . Developer survey results 2018. URL: https://insights.stackoverflow.com/survey/2018#technology-programming-scripting-and-markup-languages.

[23] Thomas, A., Gupta, A., 2016. Market guide for application platforms. URL: https://www.gartner.com/doc/3522917/market-guide-application-platforms.

[24] Vaishnavi, V., Kuechler, W., 2004. Design research in information systems. URL: http://www.desrist.org/design-research-in-information-systems/.

[25] WildFly, . Wildfly documentation. URL: http://docs.wildfly.org/16/.