Comparative Analysis of EJB3 and Spring Framework

Janis Graudins, Larissa Zaitseva

Abstract: *The paper describes main facilities of EJB3 and Spring Framework as well as the results of their comparative analysis. Basic features of EJB3 and Spring Framework used for business component management are outlined. The results of frameworks investigation and evaluation are considered. For this purpose 13 criteria have been selected. Offered criteria allow selecting an appropriate framework for definite requirements satisfaction.*

Key words: *Spring framework, Enterprise JavaBeans 3.0, Java2 Enterprise Edition, Business components.*

INTRODUCTION

Applications based on Java programming language take a significant part of developed software. Java was initially created by Sun Corporation has strong open-source community encouragement and extensively supported by IT giants like BEA Systems, IBM Corporation and JBoss. Java is especially popular in large enterprise application development. IDC market research has shown that 25.3% of surveyed large companies use Java for their most important applications (October 2005 report) [7]. Java 2 Enterprise Edition (J2EE) platform, which provides great opportunities for distribute systems development, is used for the most enterprise applications and Enterprise JavaBeans (EJB) technology is a heart of it. Usually architecture of J2EE application contains several separate layers (Fig. 1). Server layer typically contains server components with application business logic, which are managed by EJB container (EJB specification implementation). EJB container is a part of the application server (typically EJB container and application server cannot be separated and are produced by the same vendor). It provides server component lifecycle, transaction and security management services.

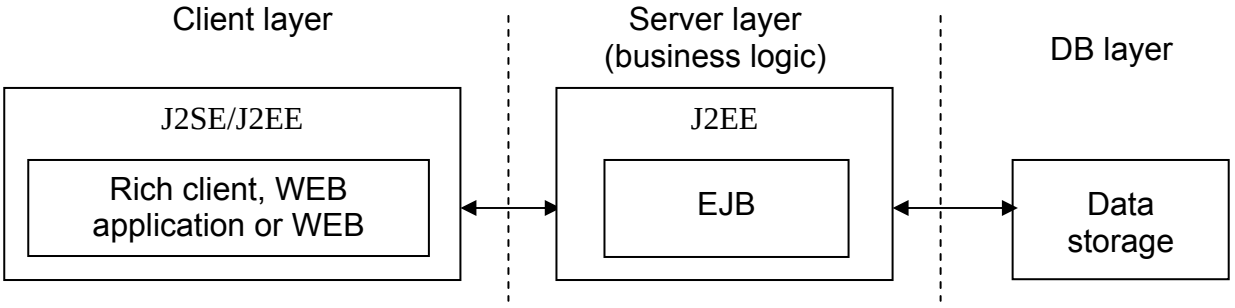

Figure 1. Classic 3-tier architecture

Unfortunately, earlier versions of EJB were too complicated and new business component management technology appeared. Spring Framework, which version 1.0 released in March 2004 [9], is a free open-source lightweight business components container that can be used with and instead of EJB. In general, Spring Framework provides some additional services like Spring Web Model View Controller (MVC), but they are out of scope of this paper.

In the beginning let us briefly describe some basic features of EJB and Spring Framework.

ENTERPRISE JAVABEANS

One of the EJB architecture goals is make it easy to write distributed object-oriented business applications in the Java programming language [1]. Unfortunately, EJB versions 1.0 – 2.1 were too complicated and did not achieve this goal. The purpose of the EJB 3.0 release is to improve the EJB architecture by reducing its complexity from the enterprise

application developer's point of view [2]. To simplify EJB architecture the following changes were done [3]:

- metadata annotations introduced in [10] can be used in combination or instead of deployment descriptor to annotate EJB applications (specify component types, behavior, etc.) as well as encapsulate environmental dependencies and resources;
- elimination of requirement for the specification of home and component interfaces;
- elimination of requirement for enterprise beans to implement specific interface (javax.ejb.EnterpriseBean);
- simplification of enterprise bean types (entity beans removed);
- interceptor facility replaced requirement for the implementation of callback interfaces;
- default values are used whenever possible ("configuration by exception" approach);
- reduction of the requirements for usage of checked exception.

Key features introduced in EJB 3.0 together with metadata annotations and interceptors are the following:

- entity persistence was simplified and support for light-weight domain modeling provided (Now it is possible to provide EJB 3.0 light-weight containers , that can be used on client layer out of the application server box);
- enhancements to EJB QL and support for native SQL queries;
- EJB container-managed timer service provided, which allows executing enterprise beans on specific time events.

At this moment EJB 3.0 specification still is in Proposed Final Draft status. It means that its implementations cannot be fully completed by EJB container vendors until Final Release issued.

SPRING FRAMEWORK

The main goal of Spring Framework producers was to create a simple alternative to EJB. The simplification of the process of application development and testing is the key goal of Spring. Framework is primary based on two core features: inversion of control (IoC) and aspect-oriented programming [11].

Usually, objects obtain references to required objects by themselves (like in EJB 2.0, bean retrieves needed resources using JNDI). Inversion of control allows injecting all dependencies into bean in its creation time by some external manager. Bean is only required to define required property in code and its mutator method (set() method). Primary source for dependency injection is xml configuration file. For example productService needs to perform some customerService operation. Then reference to customerService will be injected in customer property of com.article.ProductServiceImpl (Fig. 2).

```
<beans>
<bean id="customerService" class="com.article.CustomerServiceImpl"/>
<bean id="productService" class="com.article.ProductServiceImpl">
<property name="customer" ref="customerService"/>
</bean>
</beans>
```

Figure 2. Dependecy injection

Aspect-Oriented Programming (AOP) allows implementing more common services (like transaction, security management, logging and etc.), which should be applied to multiple components. In case of AOP usage component does not have any knowledge that

it is wrapped by some services. AOP used in Spring [8] (In EJB 3.0 AOP can be used through Interceptors):

- To provide declarative enterprise services (e. g. declarative transaction management);
- To allow users to implement custom aspects.

Spring provides a number of additional services, which are based on IoC and AOP key features. These services should be compared with EJB services to make overall frameworks evaluation. To compare both EJB 3 and Spring Framework a set of criteria is offered.

COMPARING CRITERIA

The purpose of comparison is to show distinction between EJB3 and Spring Framework. To achieve this goal the following criteria were selected:

1.) Transaction manager allows comparing kinds of supported transactions implementations.
2.) Transaction opportunities criterion includes transaction attributes support, isolation levels, flat or nested transaction support.
3.) Entity persistence helps to evaluate provided functionality for persistent objects, Object-Relational Mappings (ORM).
4.) AOP (Interceptors) shows provided functionality for aspect-oriented programming.
5.) Application configuration – possibility to setup applications configuration and declarative services.
6.) Security allows comparing provided security level and services.
7.) Services flexibility evaluates opportunity to replace services, wire required services.
8.) Services integration detects integration opportunities, especially with application servers.
9.) Additional functionality describes additional services provided by framework.
10.) Testability criterion is used for evaluation of testing, its simplicity and opportunity to test all components
11.) Technology maturity, support. This criterion shows how mature is product and companies that support it.
12.) Price – possible price of product (for EJB based on previous investigations in [4]).
13.) Documentation – provided documentation, examples and support.

Each criterion was evaluated in range from 0.0 (do not support) to 1.0 (completely support). The results of EJB and Spring Framework evaluation are shown in Table 1.

Table 1

EJB and Spring evaluation by criteria

Eval.	EJB 3.0	Spring	Eval.
0.7	**1. Transaction manager**		**0.9**
Only JTATransactionManager can be used. This is the primary manager used by business applications and the only one that could be used if application works with 2 (or more) data sources		JTATransactionManager as well as selected ORM provider transaction manager can be used (Hibernate, JDO, JDBC, OJB)	
0.6	**2. Transaction opportunities**		**0.8**
Only transaction attributes are supported, transaction cannot be nested		Supports transaction attributes as well as isolation levels, nested transaction supported if transaction manager support them	
0.9	**3. Entity persistence**		**0.7**
Own entity manager defined, possibility to use annotations in ORM, EJB QL and native SQL		Third-party ORM implementations like Hibernate, JDO, iBATIS, OJB	

support, integration with Hibernate		
1.0	**4. AOP (Interceptors)**	**0.9**
Default interceptors could be specified (apply to all components), callback interceptors. Interceptors could be implemented in the same or separate class. Could be set using annotations and deployment descriptor	Provide declarative enterprise services, custom aspects could be defined	
1.0	**5. Application configuration**	**0.8**
Primarily use metadata annotations, but it is possible to override them in deployment descriptor	Primarily use XML configuration file, possible to use Jakarta Commons Attributes or standard J2SE 5.0 annotations	
0.9	**6. Security**	**0.6**
Supports declarative security through metadata annotations and declarations in deployment descriptor	Provides integration with open source Acegi security framework, which supports declarative security and based on IoC and AOP usage	
0.7	**7. Services flexibility**	**1.0**
Depends on EJB implementation. If server provides modular structure, then only required services can be used	Any services can be assembled, using xml configuration file	
0.9	**8. Services integration**	**0.7**
Application server contains implementation of EJB and it gives an opportunity to optimize performance, clustering support	Spring framework is created separately from application server and it more difficult to optimize integration. Not applicable if no application server is used.	
N/A	**9. Additional functionality**	**N/A**
Depends on EJB implementation (server provider)	Provides integration opportunities with various open-source products, Spring MVC	
0.8	**10. Testability**	**1.0**
Most components are testable outside container, but container service object should be tested inside container (for example EntityManager)	All component are testable outside container, IoC allows to use mock object for testing purposes	
0.7	**11. Technology maturity, support**	**0.6**
EJB is standard, which created by experts from different vendors (including Oracle, BEA, IBM), all its versions are fully compliant. Only in PFD status, no valid implementations	Open-source technology primarily supported by Interface21. Is rather mature (2 years since release 1.0), but not a standard	
0.3	**12. Price**	**0.9**
Primarily paid product (free JBoss EJB implementation)	Free open-source product	
0.8	**13. Documentation**	**0.5**
Detailed EJB specification as well as application server documentation is provided. Various examples provided. Support from vendors	Documentation and Javadoc do not contain all technical details, too few examples	

Results of the comparison show that Spring framework is especially preferable for using in small companies, which deal with various open-source products. It is very simple, convenient and flexible framework, but very powerful in the same time. We can recommend to use Spring in cases, when heavyweight component container is not needed.

From the other side EJB 3 can be useful for companies, which plan to maintain long-term applications based on EJB. This technology will be supported for a long time and new versions will be always compatible with the old ones. EJB container integration with application server provides great opportunities in scalable, highly optimized program development.

CONCLUSIONS AND FUTURE WORK

The comparative analysis of EJB and Spring Framework has led to the following conclusions:

- offered criteria and results of evaluation can be useful for IT companies, to assist in business component management framework selection and usage;
- selected criteria are flexible, so it can be extended accordingly frameworks evolution;
- multiattribute method, introduced in [5], [6] can be used to made an overall framework evaluation accordingly to company's requirements.

In future: 1) Results of EJB3 evaluation will be re-examined and possibly changed after final release is issued; 2) more detailed comparison should be done between EJB ORM implementation and Hibernate (as primary ORM framework for Spring).

REFERENCES

[1] L. DeMichiel, "Enterprise JavaBeans Specification, Version 2.1", [Online document], Sun Microsystems, November 12, 2003, 646 pages, Available at HTTP: http://java.sun.com/products/ejb/docs.html

[2] L. DeMichiel, M. Keith "JSR 220: Enterprise JavaBeans, Version 3.0. EJB 3.0 Simplified API", [Online document], Sun Microsystems, December 18, 2005, 59 pages, Available at HTTP: http://jcp.org/aboutJava/communityprocess/pfd/jsr220/index.html

[3] L. DeMichiel, M. Keith "JSR 220: Enterprise JavaBeans, Version 3.0. EJB Core Contracts and Requirements", [Online document], Sun Microsystems, December 18, 2005, 526 pages, Available at HTTP: http://jcp.org/aboutJava/communityprocess/pfd/jsr220/index.html

[4] J. Graudins "Comparing analysis of Java application servers", Scientific proceedings of Riga Technical University, 2004, p. 118 – 125.

[5] J. Graudins, L. Zaitseva "Application Server Evaluation Method", Proceedings of the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing" (CompSysTech'05), 2005, p.IIIB.6-1 – IIIB.6-6.

[6] J. Graudins, L. Zaitseva "Application Server Selection for definite systems' class", Proceedings of 19th International conference "Systems" for Automation of Engineering and Research" (SAER-2005), 2005, p.230 – 235.

[7] S. Hamm, "Java? It's So Nineties", [Online document], BusinessWeek Online, December 13, 2005, Available at HTTP: http://www.businessweek.com/technology/content/dec2005/tc20051213_042973.htm

[8] R. Johnson, J. Hoeller, A. Arendsen "Spring. Java/J2EE Application Framework", [Online document], 2004-2005, 290 pages, Available at HTTP: http://www.springframework.org/documentation

[9] R. Lambert, "An Introduction to the Spring Framework", [Online document], Chicago Java Users Group, June 21, 2005, Available at HTTP: http://cjug.org/presentations/2005/June21/Spring-Framework-Intro-Rob-Lambert.ppt

[10] R. Mordani "Common Annotations for the Java Platform", [Online document], Sun Microsystems, October 12, 2005, 32 pages, Available at HTTP: http://jcp.org/aboutJava/communityprocess/pfd/jsr250/index.html

[11] C. Walls, R. Breidenbach "Spring in Action", Manning Publications, 2005, 472 pages.

ABOUT THE AUTHORS

Janis Graudins Mr. sc. ing., PhD student, Sun Certified Business Component Developer for Java 2 Enterprise Edition, Software Engineering Department, Riga Technical University, Phone: +371 641543, E-mail: johnyk23@inbox.lv.

Prof. Larissa Zaitseva, Dr. sc. ing., Software Engineering Department, Riga Technical University, Phone: +371 7089571, E-mail: lzaiceva@egle.cs.rtu.lv.