A Study of Software Architecture and Comparative Performance Analysis for an EJB-Based Flexible Enterprise Application Framework

Yonghwan Lee[1]
Department of Electronic Engineering and Computer Science, School of Engineering
Vanderbilt University, Nashville, TN 37232 USA
ylee@isis.vanderbilt.edu

Dugki Min
School of Computer Science and Engineering, Konkuk University, Hwayang-dong,
Kwangjin-gu, Seoul, 133-701, Korea
dkmin@konkuk.ac.kr

Junaid Ahsenali Chaudhry, Seungkyu Park
Graduate School of Information and Communication, Ajou University,
Woncheon-dong, Paldal-gu, Suwon, 443-749, Korea
{Junaid, sparky} @ajou.ac.kr

Abstract

We observe that the e-business systems development frameworks tradeoff performance at the expense of flexibility. In this paper, we present a performance comparison of JavaBeans application framework with a well-known framework, Struts. JavaBeans is a flexible and extensible CBD application framework. However the flexibility and extensibility are conflicting software qualities against the performance. Our experiment results show the significance of JavaBeans application framework over contemporary CBD application frameworks and how much its performance is affected by changing schemes of the framework for achieving flexibility and extensibility.

1. Introduction

The World Wide Web (WWW) contents are being updated extensively everyday. One of the major goals of complex and changeable e-business projects is to develop e-business applications fast and effectively, which not only satisfy given functional requirements, but also handle frequent changes of their requirements [1, 2]. In this fast changing environment, the most desired characteristics among e-business applications are less complexity and highly flexibility. For this purpose, many e-business development projects employ very flexible and extensible application frameworks that produce high development productivity with high software qualities such as a performance [15, 16].

An application framework is a semi-application [3, 4] of which some parts may be changed or reused. There are four popular web application frameworks, Velocity [9], Struts [10], Spring [11], and Hibernate [12]. The Velocity is a framework for rendering data in the

[1] This research was funded in part by the Team for Research in Ubiquitous Secure Technologies (TRUST) NSF CCF-0424422, and NSF S&T Center

presentation tier. The Struts is an extensible web application framework designed in the MVC architecture pattern [13, 14]. It is also a presentation-tier framework, but does not cover the business logic tier. The Spring is mainly responsible for managing objects in business logic tier. It uses a layered architecture pattern and is good for test-based projects. The Spring provides infrastructure services required in application development. The Hibernate is a framework for mapping an object with its relation table. It provides association, composition, inheritance, and polymorphism relationships. In addition, it provides powerful query described by the Hibernate query language. However these four frameworks intend to a specific target tier and also do not provide systematic integration through all of the tiers. Moreover in order to cope with frequent changes of the functional and quality requirements, semi-application frameworks need to solve some of design issues, such as flexibility, extensibility, dynamic reconfiguration and management of various resources [5, 6, 7]. To address those problems, we have proposed a flexible and extensible CBD application framework, called JBean [8]. JBean has been used a number of large Korean e-business projects with its high productivity and maintainability.

In this paper, we present the performance comparison of JBean application framework with a well-known framework, Struts. Flexibility and performance are two software qualities that general CBD application frameworks intend to achieve, but in trade-off. In other words, if we try to put emphasis on a flexible software quality without considering a performance software quality, the software can be dynamically changed but it can't be used since it has too low performance. So when we develop an EJB-based flexible distributed system, we have to consider how to achieve the balance of two conflicting software qualities such as a flexibility and performance. In this paper, we show two kinds of performance experiment results. First is the significance of JBean framework over contemporary CBD application framework. Second is how much its performance is affected by changing schemes of the framework for achieving flexibility and extensibility.

The remainder of this paper is organized as follows. In section 2, we present the overall architecture of JBean framework. Section 3 explains the flexibility aspects of the framework and section 4 shows the experimental results of analyzing performance of the JBean framework with the Struts. In section 5 conclude the paper.

2. Software Architecture of JBean Framework

Figure 1 shows the overall architecture of JBean framework, proposed in [8]. The framework architecture is composed of three major subsystems: Presentation Tier, Business Tier, and Admin Console. The Presentation Tier accepts requests through client browsers, processes session management, security, and data translation, and transfers the requests to EJB-module that contains business logics. The Business Tier contains business logics to process the requests from the Presentation Tier with the help of EJB-module. The Admin Console has the development tool and management tool.

In the Presentation Tier, the FrontServlet keeps a number of servlets, and proceeds client requests according to the requesting URL pattern, such as *.page, *.do, *.admin, *.login, etc. As for a *.do URL pattern, the FrontServlet makes the EJB Servlet process a business logic. As for a *.page URL pattern, it makes the PageServlet process a UI task according to the page construction information, which is organized by the Admin Console. The Action Processor is in charge of processing action objects plugged into the framework. The action object is

generated from the framework after a developer develops an action class according to the hook method defined by the framework for processing business logic of the presentation tier.

In the Business Tier, the EJBDelegator communicates with EJBs for clients' sake. The transmitted communication information and invocation information are set in the Admin Console, because the Admin Console contains all the information and parameters classified. The FaçadeDelegator is the entry point for the requests from the presentation tier. Its major responsibility is to call the façade bean for invoking an EJB component containing appropriate business logic. The FaçadeDelegator also performs general-purpose tasks, such as exception handling and logging, which is independent of a specific subsystem. The FaçadeBean provides interface that can be used outside of the Business Tier to invoke the JobBean that has the actual business logic. The CMP/DAO processes the relational database tasks.

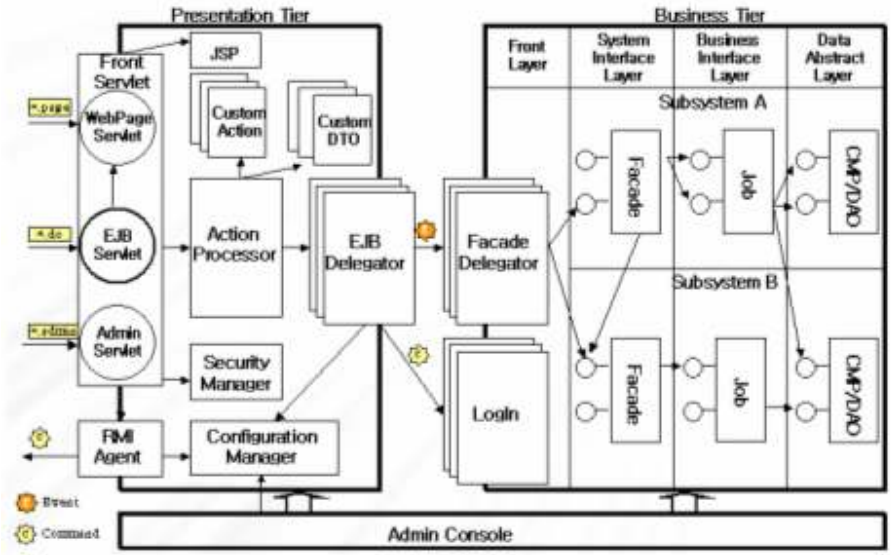

Figure 1. the Overall Software Architecture of JBean Framework

The Admin Console is a subsystem that is in charge of managing source codes and various parameters according to the concerned task unit. When a client request arrives into the Presentation Tier, the FrontSevlet in the Presentation Tier parses the client request and makes the Admin Console process the client request according to the client URL patterns. The client URL has a category information classified by the Admin Console and a URL pattern. For example, when a client requests http://localhost/aaa.bbb.do , which is a *.do URL pattern, the Admin Console develops the bbb node under the aaa node in a tree form and provides the related source code to perform the task in that node. In other word, the Admin Console categories tasks in a tree form, provides the related source code (DTO, Action, EJB code, etc.) to the task, edits them, compiles them, packages them, distributes them, and tests them via a tool. The Admin Console also manages setting information that is necessary for each task.

3. Achieving the Flexibility

Among a number of software architectural qualities, our framework provides a good flexibility in many aspects. In this section, we explain the flexibility aspect of our framework.

3.1 Flexible Control

The overall software architecture follows the MVC pattern in all of the Business Tier, the Presentation Tier, and the Admin Console. Figure 2 shows the control flow among MVC's model, view, and controller by using the Admin Console. The EJB Bean acts as a MVC's model role by containing business logics. The FrontServlet is in charge of flow control for the PageServlet to assemble JSP pages according to the Composite Pattern. The Admin Console has all the setting values for this flow control and page decision in the hierarchical form of task nodes and the next URL properties. If the next URL pattern is in *.page form, the control moves to WebPage Servlet after processing the business logic. If the next URL pattern is in *.do form, the control moves to the EJB Servlet with a different URL by means of redirection. Since the Admin Console has the setting values and parameters to control the flow, it gives a flexible control flow via a MVC's model and view.

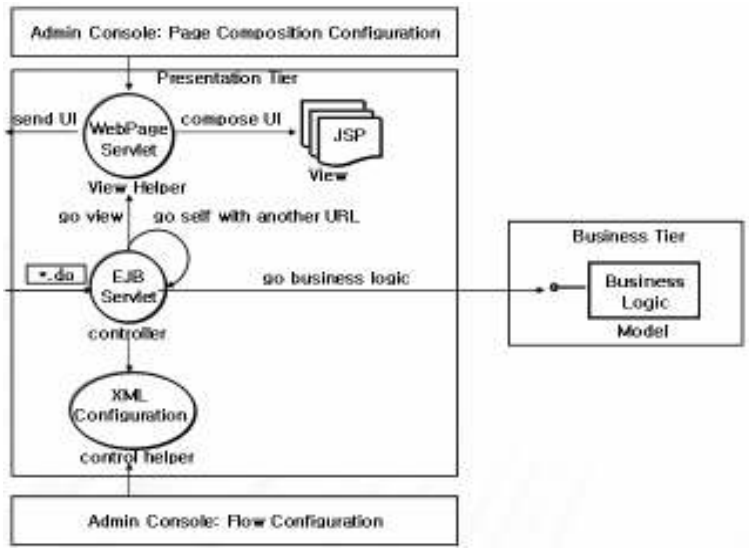

Figure 2. Flexible Control of MVC's Flow by Using the Admin Console

3.2 Flexible Maintenance

In order to maintain the framework, it is necessary to update, compile, package, distribute, and test source codes and setting parameters without stopping the server. Figure 3 shows the structure of flexible maintenance by choosing the related components in the framework. The user can update the setting parameters and source codes through the Admin Console. If there are any changes in setting of the Admin Console, the information in the XML Repository is changed accordingly by the Configuration Manager and the information in memory is changed by the XML Controller. These changes dynamically affect the processing because the Front Servlet, Action Processor, and EJB Delegator retrieve the related information from

the Configuration Manager. Thus, without stopping the server, we are able to test the source codes that are added by developer during operation, which are generally the codes for action, DTO, EJBs, etc. Those kinds of codes are compiled and added into the Object Pool with the help of the Pool Manager.

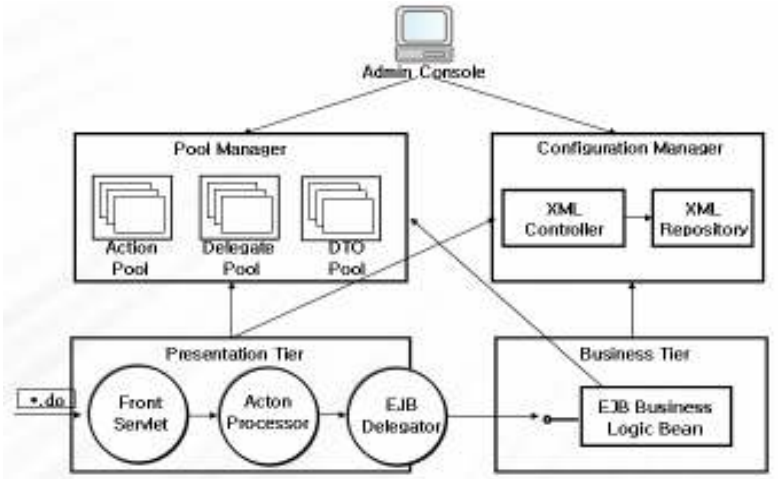

Figure 3. Flexible Maintenance of Source Codes and Configuration by using Configuration Manager and Admin Console

3.3 Decoupling between GUIs and Business Logics

There might be always changes in the GUI level. By decoupling between the GUIs of the Presentation Tier and business logics of the Business Tier, the changes of the GUI level do not affect the EJBs of business logics. In order to minimize the affection, we separate a Custom DTO from a Domain DTO. The Custom DTO generally contains data to transfer to Presentation Tier and the Domain DTO has EJB business logics. This separation can minimize the affection of GUI changes and it can yield the high flexibility of EJB-based enterprise application framework.

3.4 Separate Development of the Presentation and the Business Logics

Our framework is designed to have minimized coupling between the Presentation Tier and the Business Tier so that the development of business logics can be fully separated from the presentation parts. As shown in Figure 4, the Presentation Tier is dependent on the Business Tier only by message invocations from the EJBDelegator to the Façade Delegator. By hiding all EJBs and providing only the interfaces of Façade Delegator, those two tiers are minimally related. When transmitting data to the FaçadeDelegator, the Event Object is used to carry data. In the Event Object, there are a number of contents. The request DTO is the data required when business logics in EJB are processed. The response DTO is returned after processing business logics in EJB. The common DTO has the data that is repeatedly needed in every request. The EJB Identifier is a unique string. The Façade Delegator uses the string in order to find the interface's public method of the Façade Bean. We can separately develop and test both the Presentation Tier and the Business Logic Tier by using the Event Object. Moreover, when you want to use a commercial product instead of the proposed JBean

framework for the Presentation Tier, all that we have to do is just adding the EJB Delegator in order to invoke the Façade Delegator. That is one of the strong points of our framework.

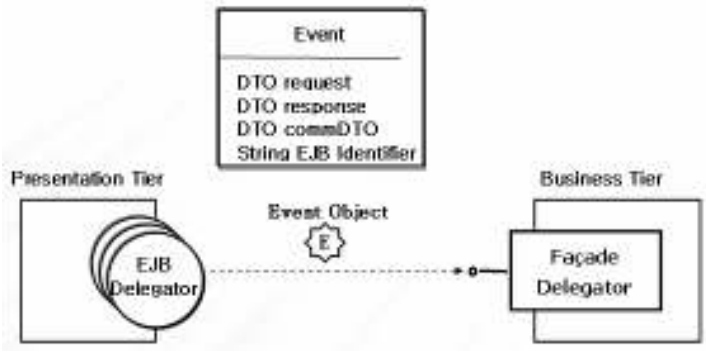

Figure 4. Decrease Dependency between the Presentation and the Business Tier by using an Event Object

3.5 Flexible Changes of Business Logic Interfaces

The interface of business logic in the Job Bean is hidden to outside of the Business Tier, so that the effect of any change can be minimized. We use the Façade Pattern, and have the interface of the Façade Bean be the multi-grained interface as shown in Figure 1. Thus, any changes in the interface of Job Bean do not affect client levels that invoke.

4. Comparative Performance Analysis

In Figure 5, we show the experimental environment for performance analysis. We use the Microsoft 2003 server for operating system, WebLogic 6.1 with SP 7 for web application server, Oracle 9i for relational database. For load generation, WebBench 5.0 tool is employed. Each job has no business logic code in order to measure the maximum throughput. TPS (Transactions per Seconds) and execution time are used for the metric of performance analysis.

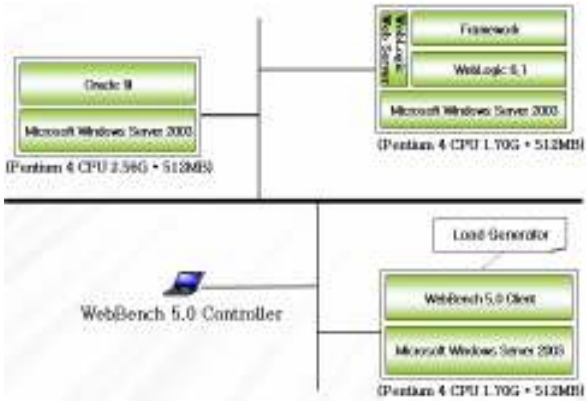

Figure 5. Experimental Environment for Performance Analysis

Table.1 shows the workload configuration for performance comparison. The performance experiment basically uses the workload template which Web-Bench 5.0 has used for testing CGI applications for e-commerce. We analyze transactional log files at the real banking sites and assign the weight of DB-related CRUD (Create, Read, Update, Delete) to 3 applications (except for the application 'log-in'). By analyzing transactional log files, we give the Read-based Application much more weight. Each Customer Action in presentation tier and Job in the business tier have no business logic code in order to measure the maximum throughput of the JBean Framework itself. The performance experiment also uses the emp table which is provided by Oracle 9i.

Table 1. Workload for Performance Comparison

Index	Application Name	URL	Weight
1	Log-In	GET /login.do	3%
2	Emp List	GET /emp.list.do	70%
3	Emp Insert	GET /emp.insert.do	10%
4	Emp Update	GET /emp.update.do	17%

In Figure 6, we show the request execution processes of four target architectures. The first two of them are variant architectures of the JBean, the third is of the Struts framework, and the last is a plain web server without using an application framework (i.e., a normal JSP). The first two JBean frameworks are different in the key components whether the PS exists or not. The variant of JBean is WebPageServlet, called PS. The PS is employed to generate the returning page easily and efficiently. Thus, in this paper, four target architectures are cases of 'No framework', 'Struts', 'JBean with PS' and, 'No PS in JBean' respectively. As the Struts framework does not provide WebPageServlet, the fourth experimental target is considered for fairness.

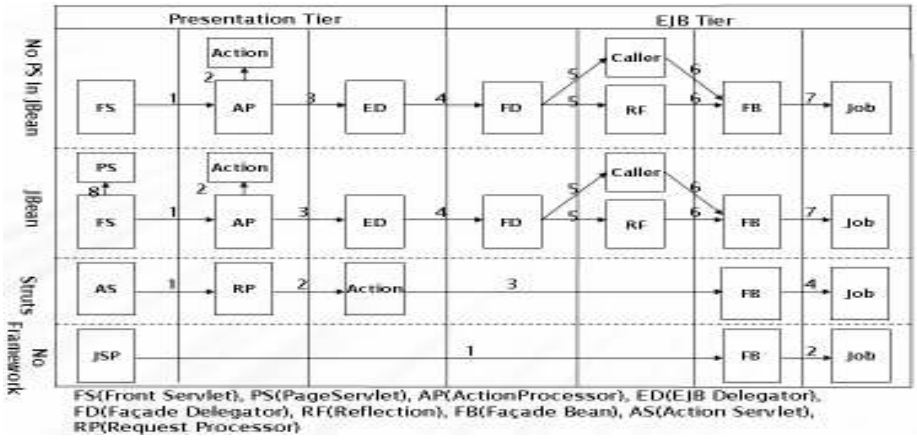

Figure 6. Four Possible Target Architectures for Performance Comparison with the Struts Framework

In Figure 7 (a), we show performance comparison of four target architectures and load analysis of main components of the framework. The first target architecture, called 'No Framework' shows 315 TPS in maximum. The second one of 'Strut's Action+ EJB' shows 280 TPS in maximum. The third one of 'JBean' shows 221 TPS in maximum. The fourth one of 'No JBean's PS' shows 253 TPS in maximum. The performance difference between the second and the third is 59 TPS, but difference between the second and the fourth is 32 TPS. Load analysis of main component in Figure 7 (b) explains the reason of performance difference between the JBean and the Struts frameworks. The JBean framework has additional processing steps as shown in Figure 3, which the Struts framework does not have. As shown in Figure 7 (b), they take additional time. Among them, WebPageServlet and object serialization for FD Invocation require more execution time.

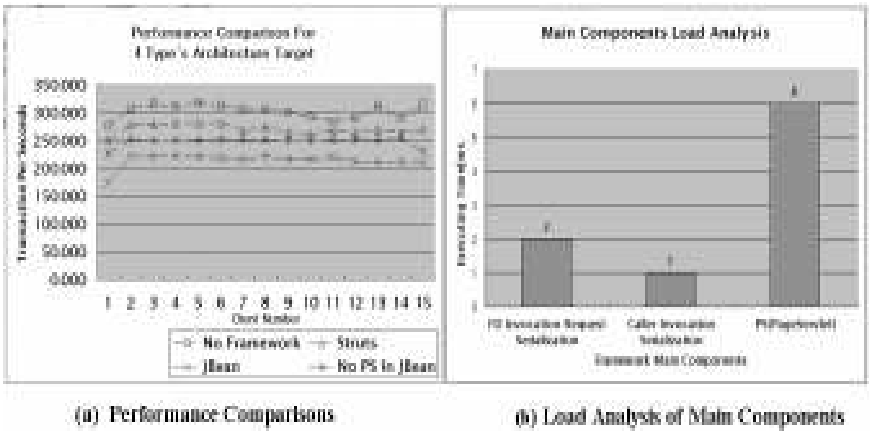

Figure 7. Four Target Architecture's Performance Comparison and Load Analysis with the Struts Framework

The Struts has just only presentation-based framework and has no business logic framework. However, JBean framework has both presentation and business logic framework. For fairness performance experiment, we must compare two frameworks in only presentation tier. So, In Figure 8, we show the performance comparison of JBean and Struts in presentation tier only, without EJB business logic tier.

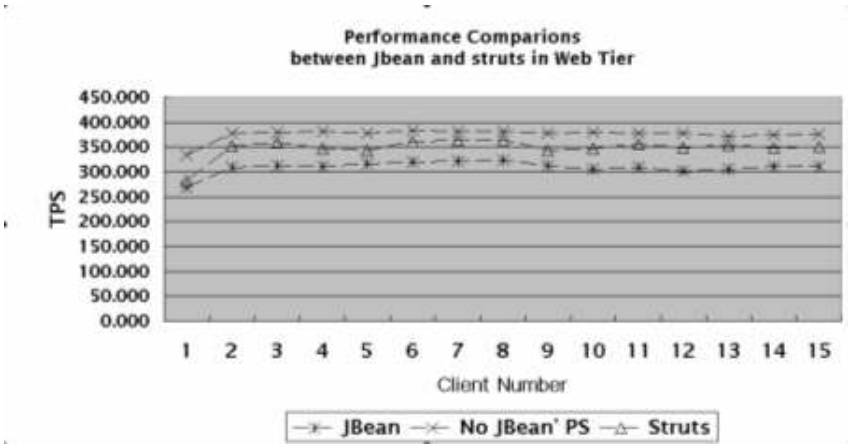

Figure 8. Performance Comparisons of JBean and Struts in Web Tier

The Struts framework can process more transactions than the JBean framework with PS. However, the JBean framework without PS can process 18 more transactions per seconds than the Struts framework. The JBean framework spends much time for processing WebPageServlet, while the Struts framework spends time in locale processing for multi-languages.

In JBean framework, when the Façade Delegator receives the request message from the presentation tier, it needs to select one of multiple Façade Beans of each subsystem and invokes it. Because there is 1: M relation between the Façade Delegator and the Façade Bean, the Façade Delegator can use one of four schemes of Figure 9 (a) to invoke the Façade Bean.

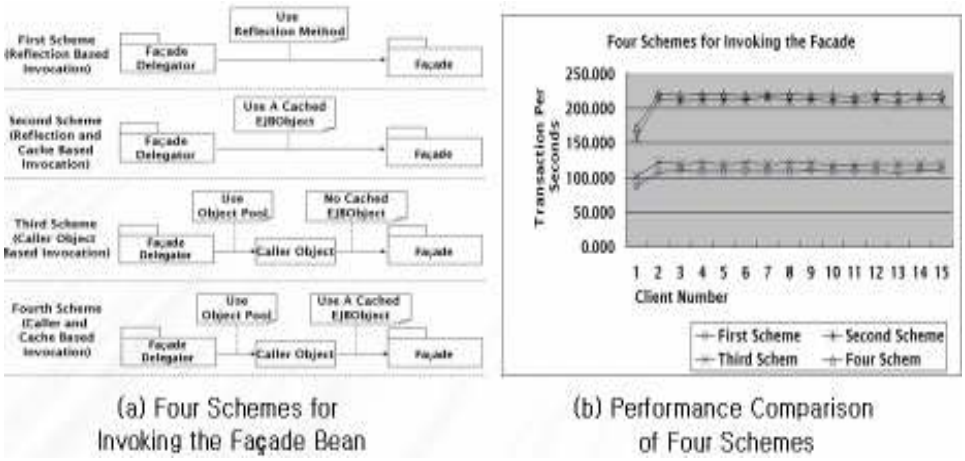

Figure 9. Four Schemes for Invoking the Façade Bean and Performance Comparison of Four Schemes

The first scheme is that the FaçadeDelegator uses the EJBMetaData and a reflection scheme without caching the reference of the EJBObject. The second scheme seems to be similar to the first. However, it is different from the first in that it uses the cashed reference of the EJBObject. The third scheme uses a callback method. The FaçadeDelegator refers the caller object with the callback method and calls the method. The callback method uses the EJBMetaData and a reflection scheme without using the cached reference of the EJBObject. The fourth scheme seems to be similar to the third. However, it does not use the cached reference of EJBObject. In Figure 9 (b), we show the performance comparison of four schemes.

In view of performance, the second and the fourth can process the more transaction per second than the first and the third. The reason is that they use the cached reference of the EJBObject. However, whenever the Façade Bean is newly created and updated, the second and the third need to restart the application server to refresh the cached reference. So, in view of flexibility, the first and third have much more flexibility than the second and the fourth. Although both schemes use the cached reference of the EJBObject, the callback scheme can process slight more transactions than the reflection. However, the callback scheme needs the additional code of development of the caller object and that of deployment to the object pool. In JBean framework, the quality manager can configure one of four schemes in the Admin Console per each task. As the first and the third has the higher flexibility than the second and

the fourth, the first and third can generally be used during the development of EJB components and the second and the fourth can be used for high performance during service.

JBean framework uses the mechanism of object pool for improving performance and flexibility. The Action, DTO, EJB Delegator and Caller object are developed in the Admin Console and deployed to the object pool. Although they are frequently updated, application server does not need to be restarted for refreshing them. In Figure 10 (a), we show the performance comparison of existence and non-existence of object pool. The object pool scheme for the Action and DTO object can process 139 transactions per seconds than no object pool scheme. In Figure 10 (b), we show load comparison of the main module with and without object pool. Because the Action Processor is responsible for creating the Action and DTO, it has the heavier load than any other main module of the framework.

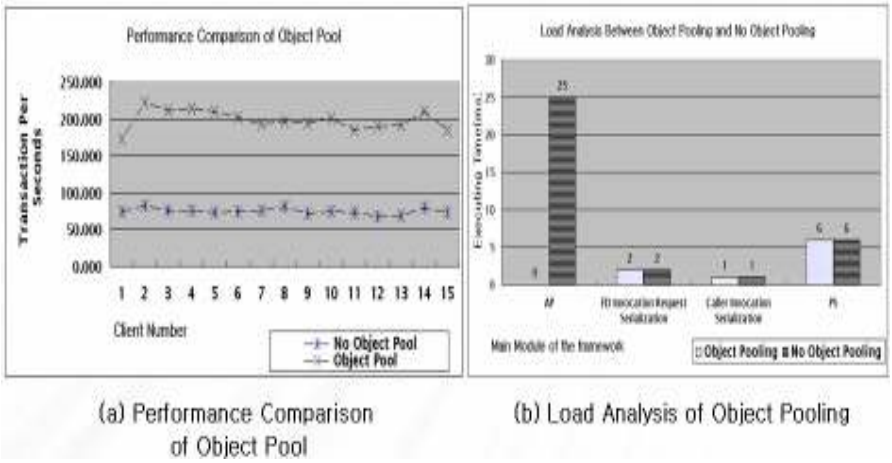

Figure 10. Performance Comparison of Existence and Nonexistence of Object Pool and Load Analysis of Object Pooling

The JBean framework provides presentation tier framework and additional business tier framework in comparison with Struts framework. So, component-based development projects can easily build applications without complexly mixing up application frameworks per each tier. Moreover, JBean framework has the additional the WebPageServlet that process each UI task according to the page construction information, which is organized by the Admin Console. Even if the JBean framework has additional processing steps, such as the WebPageServlet in presentation tier, the difference of performance between JBean and Struts framework is slight. The JBean framework without the WebPageServlet can process 18 more transactions per seconds than the Struts framework. In analyzing the performance of the JBean framework, we show how much its performance is affected by changing schemes (i.e., object pooling and EJB invocation method) of the framework for achieving flexibility. Moreover, because JBean framework categories tasks in a tree form in the Admin Console, the framework can change dynamically the schemes that must be considered when handling conflicting qualities, such as performance and flexibility.

5. Conclusion

Many e-business development projects employ very flexible and extensible application frameworks that produce high development productivity with high software quality. In this

purpose, we have proposed an flexible and extensible CBD application framework, called JBean. To balance flexibility and performance qualities, which are in trade-off, many CBD application projects need to analyze the performance of the employed CBD application framework. In this paper, we show the performance comparison of JBean framework with a well-known framework, Struts. JBean framework provides presentation tier framework and additional business tier framework in comparison with Struts framework. So, component-based development projects can easily build applications without complexly mixing up application frameworks per each tier. The JBean framework without PS can process 18 more transactions per seconds than the Struts framework. The JBean framework spends much time for processing WebPageServlet, while the Struts framework spends time in locale processing for multi-languages. When the Façade Delegator invokes the Façade Bean, there are four schemes in JBean framework. The callback scheme is slight better than the reflection scheme. In analyzing the performance of JBean framework, we show how much its performance is affected by changing schemes (i.e., object pooling and EJB invocation method) of the framework for achieving flexibility. Although flexibility and performance is qualities in trade-off, the object pool scheme has many advantages for both flexibility and performance. Moreover, because JBean framework categories tasks in a tree form in the Admin Console, the framework has the dynamic changeability of the schemes that must be considered in view of conflicting qualities, such as performance and flexibility.

References

[1] D. Schwabe and G. Rossi, "An Object-Oriented Approach to Web-Based Application Design," Theory and Practice of Object Systems (TAPOS), special issue on the Internet, vol. 4, no. 4, 1998, pp. 207-225.

[2] Sencan Sengul, James W, Gish, James F. Tremlett, Building a Service Provisioning System Using The Enterprise Java Bean Framework, IEEE, 2000.

[3] D.C. Luckham, J. Vera and S. Meldal, Key Concepts in Architecture Definition Language, in Foundations of Component-Based System, Ed. Gary T, Leavens and Murali Sitaraman, Cambridge University Press 2000.

[4] M. Fayad, D, Schemidt, and R. Johnson, eds., Building Application Framework, Wiley & Sons, New York, 1999.

[5] J. Kramer and J. Magee, Analyzing dynamic change in distributed software architectures, IEE Proceedings-Software, 145(5), Oct. 1998.

[6] D.M. Heimbigner and A.L. Wolf. Post-Deployment Configuration Management. In Proceedings of the Sixth International Workshop on Software Configuration Management, number 1167 in Lecture Notes in Computer Science, pages 272–276. Springer-Verlag, 1996.

[7] Fumihiko Kitayama, Shin-ichi Hirose, Goh Kondoh, Design of a Framework for dynamic Content Adaptation to Web-Enabled Terminals and Enterprise Applications, IEEE, 1999.

[8] Yonghwan Lee, Eunmi Choi, Dugki Min, A CBD Application Integration Framework for high Productivity and Maintainability, ICCSA 2005, pp. 858-867, 2005.

[9] Velocity, Velocity Framework, http://jakarta.apache.org/velocity/.

[10] Apache, Struts Framework, http://jakarta.apache.org/struts/index.html.

[11] Spring Framework, http://www.springframework.org/.

[12] HiberNate Framework, http://www.hibernate.org.

[13] D. Le Metayer, Describing Software Architecture Styles Using Graph Grammars, IEEE Transactions on Software Engineering, 24(7):521-533, July 1998.

[14] E. Gamma, R. Helm, R Johnson, J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley, 1995.

[15] Romi S. Wahono, Jingde Cheng: Extensible Requirements Patterns of Web Application for Efficient Web Application Development. CW 2002: 412-418

[16]Oktay Altunergil, " Easing Web Application Development with CVS", O' Reilly Network, 2002.

Authors

Name: Yonghwan Lee

Address: Department of Electronic Engineering and Computer Science, School of Engineering Vanderbilt University, Nashville, TN 37232 USA

Education & Work experience: Yonghwan Lee a Research Scientist in Department of Electronic Engineering and Computer Science, School of Engineering Vanderbilt University in USA. His current research interests include Model Integrated Computing, Component-Base Development, Software Architecture Design, Application Frameworks, and Web Service Middleware. He received the degree in public administration from Konkuk University, Korea in 1997. He received the Master and Ph.D. degrees in computer science from Konkuk University, Korea in 1999 and 2006.

Name: Dugki Min

Address: School of Computer Science and Engineering, Konkuk University, Hwayang-dong, Kwangjin-gu, Seoul, 133-701, Korea

Education & Work experience: Dugki Min was born in Seoul, Korea, in 1964. He received the degree in industrial engineering from Korea University, Korea in 1986. He received the Master and Ph.D. degrees in computer science from Michigan State University, U.S.A., in 1991 and 1995. He is currently an Associative Professor in the School of Computer Science and Engineering at Konkuk University, Korea, where he has been since 1995. His current research interests include internet distributed systems, ubiquitous and web service computing, software architecture modeling. He is the chair of technical committee in Korean Software Component Forum, and the vice-president of Korean Modeling Technology Forum.

Name: Junaid Ahsenali Chaudhry

Address: Graduate School of Information and Communication, Ajou University, Woncheon-dong, Paldal-gu, Suwon, 443-749, Korea

Education & Work experience: Junaid Ahsenali Chaudhry did his masters and PhD from Ajou University, Korea and currently based in University of Trento, Italy as a post doctoral student. His areas of interests include Autonomic Ubiquitous System Testing, Autonomic Self Healing Engine (ASHE) Development, Autonomic Service Composition vs. connectionless handover in pervasive environments, Performance Improvement in Component-Based Systems and distributed computing.

Name: Seungkyu Park

Address: Graduate School of Information and Communication, Ajou University, Woncheon-dong, Paldal-gu, Suwon, 443-749, Korea

Education & Work experience: Professor Park Seoungkyu heads the Multimedia and Computer Architecutre (CeMulti) Lab in the department of information and communication of Ajou University. His area of interest include Testing Embedded Systems, VOD media streams, Large Scale Network Simulations for Cyber Attacks, Advanced Computer Architectures.