웹계층 오브젝트 모델링을 통한 분산 애플리케이션 개발 프레임웍크

천 상 호[†] · 권 기 현[††] · 최 형 진[†††]

요 약

분산 애플리케이션을 위한 다계층 모델 또는 분산 아키텍처를 개발하기 위해서는 웹디자이너와 페이지 작성가의 역할 분리, 엔터티 정의와 사용에 대한 고려, 데이터베이스 연결 및 관리, 트랜잭션 처리 등 여러 가지 사항을 고려하여야 한다. 본 논문에서는 분산 애플리케이션 개발시 여러 고려사항에 대한 해결책으로 웹계층 오브젝트 모델링 방법을 사용하는 DONSL(Data Server of Non SQL-Query) 아키텍처를 제안한다. 이 아키텍처는 트랜잭션 처리를 지원하고 웹계층과 DBMS 사이에는 질의 로직을 단순화하는 방법을 통해 성능을 보장하는 구조이다. 제안한 개념적인 프레임웍은 각 계층(tier)의 작업을 단순화 시키고 엔터티와 DAO(Data Access Object)를 제거시킴으로서 중대규모 사이트 구현을 용이하게 하는 방법을 제공한다.

A Framework for Developing Distributed Application with Web-Tier Object Modeling

Sang-Ho Cheon[†] · Ki-Hyeon Kwon[††] · Hyung-Jin Choi[†††]

ABSTRACT

To develop multi-tier model or distributed architecture based distributed application needs to consider various aspects such as division of role between web-designer and software developer, defining entity and its usage, database connection and transaction processing etc. This paper presents DONSL(Data Server of Non SQL-Query) architecture that provides solution to above aspects through web-tier object modeling. This is the architecture that guarantees the transaction processing and performance between web-tier and DBMS through simplified usage of query logic property. This new conceptual framework also solves enterprise site implementation problems simplifying tier, and removing DAO(Data Access Object) and entity.

키워드 : 분산 아키텍처(Distributed Architecture), 소프트웨어 프레임웍크(Software Framework), 웹계층 오브젝트 모델링(Web-Tier Object Modeling)

1. Introduction

In recent days, there is significant increase in the researches related to the application of framework and pattern on system over the Internet and distributed environment[1 - 3] instead of old OOHDM[15] Model.

The need of application is for maximizing the mutual operation, composition, extendibility and reusability of the web-application. And it also makes possible to divide the once-written system into various classes, easing the system upgrading and expansion[5, 6]. But, various facilities are needed to develop a system using framework and pattern. If the ultimate goal of using framework and pattern is maximizing reusability and satisfying mutual operation, the division of work between web designer & software developer, sharing of entity within the system, automatic creation of SQL-query to connect database and transaction processing through agent should be done[5].

In this paper, instead of making improvement on existing WAS(Web Application Server), DONSL(Data Server of Non SQL-Query) architecture is designed and implemented which is more effective and has improved development productivity. Chapter 2 contains theoretical concepts and drawbacks of n-tier structure and MVC (Model View Controller)-framework. Chapter 3 proposes MVC Model based DONSL architecture and also contains the detail explanation of distributed TX spec., DONSL

† 준 회 원 : 강원대학교 대학원 컴퓨터과학과
†† 정 회 원 : 삼척대학교 정보통신공학과 교수
††† 종신회원 : 강원대학교 컴퓨터과학과 교수
논문접수 : 2004년 5월 3일, 심사완료 : 2004년 6월 28일

agent, development roles and advantages of DONSL architecture. In chapter 4, we describe the implementation of DONSL architecture and its performance. Finally, we concluded with conclusion and future research to be done for maximizing the result of DONSL architecture.

2. Software Framework and MVC Model

2.1 N-Tier Architectures(3-tier and Multi-tier, WAS Tier)

Browser, application server that is connected by servlet or CGI(Common Gateway Interface) and database store forms 3-tier architecture. Though object modeling and implementation is simple, but it lacks the solution to complex web application required by any enterprises. And, the processing time also increases as the user number increases[4].

Multi-tier architecture-WAS tier satisfies the complex requirements and improves the multi-user processing speed. And, it also separates clearly the presentation part and business logic part. But it requires the detail understanding of distributed architecture making object modeling complicate. And, it holds high chances of error occurrences which can lengthen development time, and can make maintenance handy.

2.2 MVC Framework & Entity : Roles and Weaknesses

MVC(Model View Controller) framework is the structure that minimizes the alteration effects through function encapsulation[11, 16]. Hence, it is accepted as a nice method in the perspective of system expansion and maintenance. Model is expressed in terms of entity in distributed system. And, entity is very important factor since all other entity related items must be modified if entity needs modification. It takes long time while projecting object as entity at the time of object-oriented analysis and designing. There is also difficulty to implement presentation entity and DBMS entity separately.

These problems tighten the coupling between tiers and become obstacle on the system development and maintenance. Since SQL query is always dependent on the database, the logical part of SQL query DAO entity needs to be considered. The changes in SQL query logic, entity and database affect all tiers. If the developer can work on any system development without worrying much about entity and DAO, complicated web application can be turned into simple.

2.3 MVC Model based Object Modeling

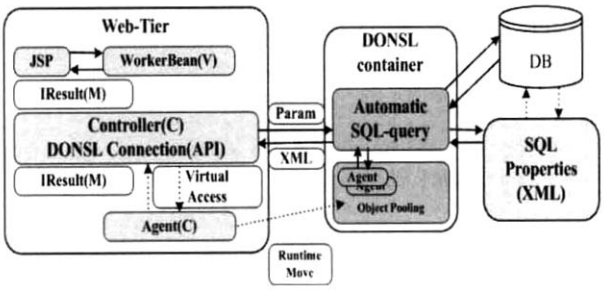
This architecture applies MVC concepts to web-tier in WAS using entity as model, WorkBean as view and Controller bean as controller. This structure does modeling only about web-tier where as WAS tier is handled by automatic transaction processing. And, the entity between each tier can remove tight coupling problem among tiers through common API.

3. DONSL Architecture based on MVC Model

3.1 DONSL Architecture

DONSL architecture consists web tier, WAS-tiered DONSL container and database. Web tier uses IResult entity as model, WorkerBean as view, and Controller bean and agent as Controller applying MVC concepts. DONSL container connects object pool agent and executes the automatic SQL query, and provides communication among each tier on XML basis. In this architecture, the developer develops only XML property used from JSP, WorkerBean, Connenction, Agent and Database. The advantages of this architecture are as follows :

- Object modeling is done from the web-tier
- Entity is used globalizing entity(IResult)
- SQL-query logic is created automatically (setting SQL properties)
- Default agent is used while handling flat Tx(API provides)
- Agent for complex Tx handling can be added



(Figure 1) Multi-Tier Architecture of DONSL

3.2 DONSL Distributed Transaction Specification

Transaction processing of DONSL distributed architecture is done by making accessible it through unit transaction from web-tier and by including it in primitive SQL-query or unit transaction. Automatic processing of transaction unit is done by executing agent for container forwarding encapsulized transaction from DONSL container to agent.

3.3 DONSL Agent

DONSL agent is used for processing complex transaction. DONSL container forwards the encapsulized transaction object to the agent and calls the execution method of agent.

Ex.) Source Code of Agent Interface :

```
public interface IDonslAgent extends IObject {
        public void unset( ) ;
        public IResult execute(IUserTransaction ut)
                throws SQLException ;
}
```

From above example, the return value IResult type object is transmitted to web-tier. And Agent moves to DONSL container at the run time reducing the resource load.

3.4 Admin Tool

Admin tool is the tool that manages all settings of DONSL-Tier. Web-tier setting is set making accessible through unit Tx. And setting steps are DB setting, primitive SQL-query setting, transaction setting. This tool is used as developer's reference tool for making possible one side object modeling(web-tier).

3.5 Development Roles

In DONSL architecture, web-designer and developer handle JSP part where as WorkerBean, Controller and Agent are handled by developer alone. And, developer also looks DONSL container that is connected with DB.

3.6 Advantages of DONSL

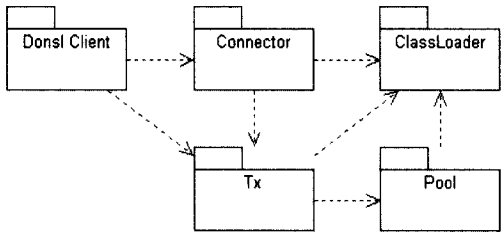
First, one side object modeling becomes possible in distributed environment. Second, it also shortens developing time making Tx processing simple and error modification easy by converting(DAO class remove) SQL-query logics into SQL-properties. Third, error modification time can be shortened by converting object transmission among tiers into XML(Entity class remove). Fourth, it supports the distributed transaction and has many other advantages like performance maximization, WAS-tier development in low cost, low maintenance cost, reduction in extra education time for existed WAS, maximization of expertness of advance technician and reduction of WAS internal structure learning time etc.

4. Prototype Implementation and Performance Evaluation

4.1 Prototype Implementation

DONSL prototype can be divided into two parts, DONSL container and DONSL client. DONSL container is again composed of 4 packages, connector package that looks communication with client, Tx package that handles transaction, ClassLoader package that looks dynamic class loading of agent. Finally, Pool package that manages database connection and object resource.

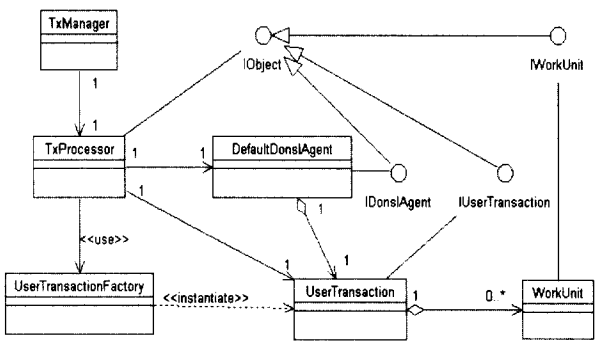
In the (Figure 2), the arrow dotted line shows the dependency relation and the most important thing is weak dependency between ClassLoader and Tx package. The prototype is designed prioritizing weak dependency package.



(Figure 2) Prototype Package

First, ClassLoader is implemented getting inheritance from J2SE API ClassLoader class since it is the weakest dependency package. Second, Tx package being the most core part of DONSL architecture manages the creation and processing of transaction. The class diagram of Tx is as follows:

TxManager handles start and end of all unit transaction as shown in (Figure 3).



(Figure 3) Class Diagram of Transaction Processing

TxProcessor encapsulates the transaction setting details of DONSL admin tool item-wise. It passes User Transaction to the agent, and calls the agent's execution method.

DefaultDonslAgent is implemented executing WorkUnit Collection that is included in UserTransaction using passed UserTransaction.

The relation between UserTransaction and WorkUnit is One to Many i.e. many jobs can be run in one transaction. In real implementation too, UserTransaction class includes WorkUnit collection.

What we have to notice from (Figure 3) is TxProcessor, DefaultDonslAgent, UserTransaction, WorkUnit all commonly get inheritance from IObject. And, these classes get inheritance from one super interface to reduce memory overload being possible to be managed from object pool.

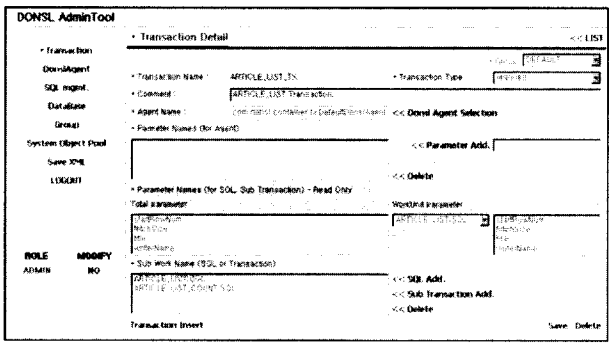
Third, pool package implemented connection pool and object pool using general cache pattern. And, connection pool manager was implemented for managing connection pool for each distributed database and object pool manager for managing object pool for each object type.

Fourth, Connector package was designed using Servlet which can execute DONSL container from any servlet container.

Fifth, since DONSL client package should communicate with Connector package, it was designed based on servlet making accessible to servlet.

Finally, DONSL admin tool was developed using Web based JSP.

Admin tool consists menu for each item as shown in the (Figure 4) below so that all items related to transaction and details of each item can be set from each menu.



(Figure 4) Admin Setting Tool

4.2 Prototype Usage Procedure

To develop projects using prototype, it is necessary to have project analysis and design at first. The followings are the setting procedure of DONSL admin tool :

First, transaction list is registered based on analysis and designing. Transaction list is made on the basis of all transactions included in the project requirements. Transaction list is registered as in the figure executing transaction menu from DONSL admin tool. Being initial record, Unique transaction name and comment should be given, and details can be added later. These settings can be a handful material while analyzing and designing any project.

Second, database is registered. Since the transaction projected from any project can be related to various databases, so the related databases should be enlisted in detail. Database list is also managed with unique name.

Third, SQL is also enlisted. The SQL details are set for accessing database as low executing module of unit transaction.

SQL settings consists unique SQL name, Objective database, input parameter and type, SQL-query. After setting the details, SQL tuning can be possible at anytime without modifying the programming code.

Fourth, unit transaction details are completed. The setting of agent, input parameter, SQL list etc is done.

In real, the automatic transaction processing of DONSL container is the execution of SQL list using set agent after getting input parameter based on transaction details. This setting can be changed anytime without caring programming code.

Fifth, when the execution method of Transaction SQL list is not simple and serial, in other word, when it is complicate, the developer defined agent must be implemented. At this time, after setting necessary agents from agent menu, Transaction details can be changed. The agent programming code can be uploaded directly or after compilation.

Finally, the DONSL server settings are completed after finishing transaction setting and saving admin tool setting details. Now, only client programming is remained which can be done almost similar like general JDBC programming. The following is the source code for accessing DONSL server from client :

```
DSURL url =
new
DSURL("donsl:http:nc:hit:hong/test@localhost:8888:ds/Dons");
DSConnection conn =
        DSConnectionFactory.getDSConnection( url, null) ;

String[]paramNames = {"COFFEE_NAME"} ;
                                    //Tool setting parameter
String[]paramValues ={"java"} ;      // Tool Parameter Name

DSStatement stmt = conn.createStatement( ) ;
stmt.setParams( paramNames, paramValues ) ;

String txName = "Test1_TX" ; //Tool setting Transaction Name :
stmt.execute(txName) ;          //Network DONSL server
IResultSetHouse house = stmt.getResultSetHouse( ) ;

IResultSet result = house.getResultSet(0) ;
while(result.next( )){
 System.out.println( result.getString("coffee_name") ) ;
 }
```

(Figure 5) Client Source Code

4.3 Performance Evaluation

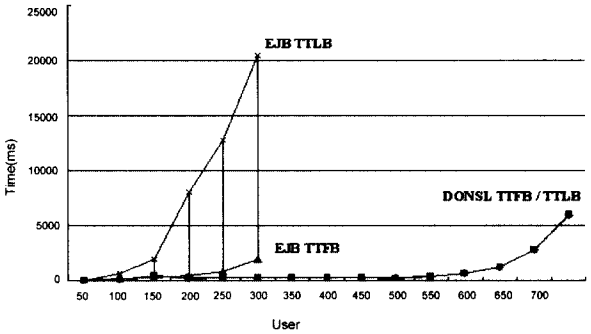
Performance evaluation was done comparing with J2EE platform Java PetStore sample application. DONSL simply changed Java PetStore sample application into DONSL server accessible format.

For making overload used in the test reasonable, the following settings are done in the Stress Tool.

- Warm up Time : 1 minute
- Measurement Time : 5 minute
- Think Time : 5~35 seconds(avg. 20 sec)

For the best result, the actual test was done taking measurements after passing warm up time and each user's think time.

Setting hardware and overload percentage properly, the test was done generating request from more than 50 virtual users till occurrence of socket error or internal server error(HTTP error code 500). Following result (Figure 6) was obtained :



(Figure 6) Response Time Comparison Graph

- TTFB(Time to First Byte) : The time after sending request and till getting first 1 byte response
- TTLB(Time to Last Byte) : The time after sending request and till getting whole

Analyzing the above graph what we proved the DONSL server yields the 5 times good performance when the users are 50 and if more, there is more significant difference.

In case of J2EE environment, the measurement was not possible due to the occurrence of socket error when there are more than 250 users.

After from this nature, the number of classes and code length per module was also measured for comparing productivity. The clear difference was seen due to use of agent in DONSL for managing complicated transaction. The following is the module class comparison table of PetStore sample application.

〈Table 1〉 Module class comparison of PetStore application

Module	Page	J2EE	DONSL
Account	Login	com.sun.j2ee.blueprints.signon.ejb.SignOnEJBcom.sun.j2ee.blueprints.signon.user.ejb.UserEJB	DEFAULT AGENT
	View Account Information	com.sun.j2ee.blueprints.customer.ejb.CustomerEJBcom.sun.j2ee.blueprints	

As seen in the above table, EJB uses module wise multiple classes where as DONSL uses user defined agent in the complicated transaction. For example, there was just one agent used for PetStore sample application. The remaining model used DefaultAgent included in the DONSL which eased the coding job. This is due to the DONSL admin tool settings.

In conclusion, it was proved that DONSL is notably far better than J2EE environment comparing performance and productivity.

5. Conclusion and Future Research

The need to improve MVC model based object modeling is growing due to the increase in the complexity of web application. We proposed new DONSL architecture based on MVC model to facilitate those concerns. However, limitations of this research should also be noted in order to motivate and guide future research.

First, we classified DONSL Architecture into Web-Tier, WAS-tiered DONSL Container and Database. Web-tier uses MVC IResult entity as model, WorkerBean as view and Agent as Container making good deal of MVC concept where as DONSL container works as a bridge among tiers. The developer only needs to develop XML property which is used from JSP, WorkerBean, Connection, Agent and Database which eases his job. And, DONSL architecture brought the efficient handling of complicated transaction through the usage of agent and hence, it was proved to be an ideal solution for complicated web application.

Though DONSL architecture provided much relief to the enterprises who were looking for the solution to present complicated web application demand of users, there is still a lot to do for improving broad acceptance of DONSL architecture and its reliability. For that, the future research should be concentrated to find the solution to following points :

- Standardization of communication protocol between DONSL client and server
- Interface for other regacy systems apart from DB
- Providing WebService of DONSL Transaction

(the WebService that communicates via HTTP/SOAP protocol for better co-ordination among enterprises)

References

[1] R. Johnson, "Frameworks = Patterns + Components," *Communication of ACM*, Vol.40, Oct., 1997.

[2] F. Bushchmann, R. Meunier, H. Rohnert, P. Sommerlad and M. Stal, "Pattern-Oriented Software Architecture A System of Patterns," *Willey and Sons*, 1996.

[3] M. Jacyntho, D. Schwabe, G. Rossi, "A Software Architecture for Structuring Complex Web Applications," *In International World Wide Web Conference(www2002)*, 2002.

[4] K. Iijima, J. Ivins, "An Alternate Three-Tiered Architecture for Improving Interoperability for Software Components," *In International World Wide Web Conference(www2003)*, 2003.

[5] F. Marinescu and E. Roman, "EJB Design Patterns Advanced Patterns, Processes and Idioms," *Wiley and Sons*, 2002.

[6] T. Fischer, J. Slater, P. Stromquist and C. Wu, "Professional Design Patterns in VB.NET Building Adaptable Applications," *Wrox*, 2002

[7] Berg, Daniel J. and Fritzinger, Steven, "Advanced Techniques for Java Developers," *Wiley and Sons*, 1998.

[8] Mowbray, Thomas J. and Ruh, William A. "Inside CORBA : Distributed Object Standards and Applications," *Addison Wesley*, 1997.

[9] S. H. Cheon, G. H. Kweon, H. J. Choi, "Developing a Automatic Components Creating System in Distributed Environment," *Korea Digial Context*, Vol.2, 2001

[10] David m. Geary, "Advanced JavaServer Pages," *Prentice Hall PTR*, 2001.

[11] Steve Burbeck, "Application Programming in SmallTalk-80 : How to use Model View Controller(MVC)," Available at http://st-www.cs.uiuc.edu/users/march/st-docs/mv.html. 1992.

[12] E. Gamma, R. Helm, R. Johnson and J. Vissides. "Design Patterns : Abstration and Reuse of Object-Oriented Design," *In European Conference on Object-Oriented Programming Processing(ECOOP'93)*, Vol.707 of Lecture Notes in Computer Science. Springer-Verlag, July, 1993.

[13] Orfali R., Hashley D. and Edwards, J. "The Essential Distributed Object Survival Guide," *Wiley*, 1996.

[14] D. C. Schmidt, "Experience using Design Patterns to Develop Reusable Object-Oriented Communication Software," *Communication of ACM(Special Issue on Object-Oriented Experiences)*, Vol.38, Oct., 1995.

[15] D.Schwabe, 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 #4, pp.207-225, Oct., 1998.

[16] G. Krasner, S. Pope, "A cookbook for using the model-view controller user interface paradigm in Smalltalk-80," *Journal of Object-Oriented Programming*, Vol.1, No.3, August/September 1988, 26-49 MVC-based Architecture for e-commerce. Journal.doc 22/22.

천 상 호

e-mail : shcheon61@empal.com

1986년 서강대학교 수학과(이학사)

2002년 강원대학교 대학원 컴퓨터과학과 (이학석사)

1985년~1989년 한국화약 전산실

1989년~1997년 (주)선경유통

1997년~현재 (주)오픈시스템서비스 대표이사

2002년~현재 강원대학교 대학원 컴퓨터과학과 박사과정

관심분야 : 컴포넌트 시스템, 분산 시스템, 미들웨어

권 기 현

e-mail : kweon@samcheok.ac.kr

1993년 강원대학교 전자계산학과(이학사)

1995년 강원대학교 대학원 전자계산학과 (이학석사)

2000년 강원대학교 대학원 컴퓨터과학과 (이학박사)

1998년~2002년 동원대학 인터넷정보과 교수

2002년~현재 삼척대학교 정보통신공학과 교수

관심분야 : 분산 시스템, 미들웨어, 임베디드 소프트웨어

최 형 진

e-mail : choihj@cc.kangwon.ac.kr

1982년 영남대학교 물리학과(이학사)

1987년 일본동경공업대학 정보공학과 (공학석사)

1990년 일본동경공업대학 정보공학과 (공학박사)

1990년~1991년 ETRI 선임연구원

1991년~현재 강원대학교 컴퓨터과학과 교수

관심분야 : 인공지능, 화상처리, 패턴인식, 컴퓨터그래픽