IMPROVING ARCHITECTURE LEVEL PERFORMANCE OF WEB APPLICATION BASED ON MIDDLEWARE TECHNOLOGY

Mani K.

Lecturer, Dhanalaksmi Srinivasan College of Engineering and Technology, Mamallapuram.

ABSTRACT

A web application is a software system that contains the computing and networking technologies required for use through web browsers on the Internet Web application. The fundamental feature of web applications is that its behavior is specified by the interaction between the environment and the system. One of the major platforms to build web applications is J2EE based on the MVC model such as Struts Framework. Struts framework based on MVC has brought the best code reuse and the legible code structure in entire system by far, but it has some problems such as complicated program, high coupling among layers and hard maintenance because its model part adopts JDBC to connect databases directly. In this paper we propose middleware technology such as Struts2 based on MVC2, Spring and Hibernate Frameworks which is the kernel and key to the simplify the software development, improve the software performance and quickly construct the Web Application of the good expansibility, maintainability. This Framework is analyzed on designing E-Commerce System.

Keywords: Middleware, MVC2, Struts2, Spring, Hibernate

I. INTRODUCTION

A. Background

In this paper, we examine a family of applications that are frequently referred to as *web applications*. Web applications play a crucial role in many organizations - to mention only systems for accounting, online shopping, securities trading, and flight reservations as examples. Business data objects are transactionally accessed by the enterprise application, and their persistent state is stored in one or more transactional data stores, for instance, relational database management systems. Most applications rely on underlying services that are commonly referred to as *middleware*. Middleware services can be regarded as an intermediate layer between applications and operating systems and typically handle distribution, heterogeneity, and transaction management for enterprise applications. Ideally, all infrastructure functionality is implemented as part of the underlying middleware, which is accessed by the Web application through standardized application programming interfaces. The use of middleware allows application developers to focus on the business logic of their web application.

B. Focus

In data-intensive applications, large numbers of business data objects are managed and accessed. Thus, efficient data management mechanisms have to be employed to access, load, store, synchronize, copy, etc. data objects in a scalable manner. In this paper, we take a middleware-centric view of data-intensive, object-oriented, multi-tiered web applications. In particular, we focus on two aspects: distribution and data management.

C. Problem Statement

Unfortunately, software architects and application developers experience two severe problems in practice:

Problem 1: With existing middleware, it is difficult to build web applications with custom distributed structures.

Typically, only a small set of standard structures is well supported by current middleware. More complex, application-specific distributed structures are either not supported or lead to significant performance problems or restrict full middleware services to a subset of distributed components only. To create efficient, custom distributed structures, application developers often have to implement typical middleware functionality (e.g., object-oriented access to data objects, transaction management, and caching) as part of their application code. This is a severe problem because implementing middleware functionality is a complex, costly, and time-consuming task that diverts application developers from the business logic.

Problem 2: With existing middleware, it is difficult to adapt the distributed structure of an existing web application to meet new requirements.

Traditionally, the distribution of web applications is considered a high-level, architectural concern. Decisions on distribution have to be made at an early design stage of a development project and are hard to change later on. Changing an existing distributed structure often requires major re-design and/or re-implementation of large parts of the web application. This is not only costly. in the worst case, this prevents organizations from quickly responding to changes.

D. Objectives

The first objective is thorough analysis will help software architects and application developers to understand what kind of problems they are facing and, in particular, why (and in which context) these problems occur.

The second objective is to develop concepts for an enterprise application middleware that

(a) explicitly supports custom distributed structures and

(b) also allows to easily adapting them to new requirements during the life cycle of an enterprise application.

The concepts can serve as a basis for developing new web application middleware as well as for extending existing middleware products and standards.

The first objective is an intermediate step on the way to achieving the second one as our main objective.

E. Approach

To address our second objective, we present our Struts2 based on MVC2, Spring and Hibernate Frameworks which realizes all of our requirements. The architecture defines principles of a middleware framework that explicitly supports custom and adaptable distributed process frameworks. This architecture relies on a network of object manager components that collectively provide data management services to Web applications. One of the main advantages of this architecture is that the underlying distributed structure of a web application can be defined and adapted through (re)configuration without having to re-design the application. This gives developers more flexibility in constructing and customers more flexibility in deploying and adapting their enterprise applications.

F. Contributions

1. The identification of requirements for Web application middleware to support custom and adaptable process frameworks along with an analysis why current middleware does not fulfill these requirements, and
2. The Struts2, Spring and Hibernate Frameworks which comprises concepts for web application middleware to support custom and adaptable process frameworks. In addition to the architecture itself, proof-of-concept implementations are provided.

G. Paper Outline

This paper is structured as follows:

Section 1 (the current section) provides a brief overview of the background, objectives, approach, and structure of the paper. In section 2, motivates the need for custom and adaptable process frameworks, identifies key requirements for such frameworks, and explains why existing middleware does not fulfill these requirements. In Section 3, we present our Middleware Technology like Struts 2, Spring and Hibernate Architecture for custom and adaptable process frameworks. Section 4 is explaining the Proof-of-Concept Implementation. We demonstrate an example web application with an adaptable custom framework, analyze performance aspects for typical scenarios, and compare our solution to an application based on object middleware. Section 5 gives a conclusion of our paper.

II. IDENTIFYING LIMITATIONS OF EXISTING MIDDLEWARE

1. No custom process frameworks.

Application developers restrict themselves to a limited subset of frameworks adequately supported by their middleware.

2. Development of a cross-process data management mechanism.

Application developers implement a full-fledged cross-process data management mechanism as part of their application code. Such solutions tend to become very complex since application developers have to deal

with many aspects that are generally regarded as infrastructure issues, for instance:

- client side caching of objects,
- managing identity of objects on the client side
- integration of cached data and client access operations into the server side transaction management,

Most application developers want to focus on business logic and are not prepared to handle these infrastructure issues, which should be part of the middleware. Even with skilled developers, this approach is costly, error-prone, and time-consuming, and thus a risk for many projects.

3. Relational Persistence for JAVA

Working with both Object-Oriented software and Relational Database is complicated task with JDBC because there is mismatch between how data is represented in objects versus relational database. So with JDBC, developer has to write code to map an object model's data representation to a relational data model and its corresponding database schema.

4. Support for Query Language

JDBC supports only native Structured Query Language (SQL). Developer has to find out the efficient way to access database, i.e. to select effective query from a number of queries to perform same task. Hibernate provides a powerful query language Hibernate Query Language (independent from type of database) that is expressed in a familiar SQL like syntax and includes full support for polymorphic queries.

5. Database Dependent Code

Application using JDBC to handle persistent data having database specific code in large amount. The code written to map table data to application objects and vice versa is actually to map table fields to object properties. As table changed or database changed then it's essential to change object structure as well as to change code written to map table-to-object/ object-to-table.

6. Optimize Performance

Caching is retention of data, usually in application to reduce disk access. Hibernate, with Transparent Persistence, cache is set to application work space. Relational tuples are moved to this cache as a result of query. It improves performance if client application reads same data many times for same write. Automatic Transparent Persistence allows the developer to concentrate more on business logic rather than this application code. With JDBC, caching is maintained by hand-coding.

III. PROPOSED MIDDLEWARE TECHNOLOGY

A. MVC2 Architecture

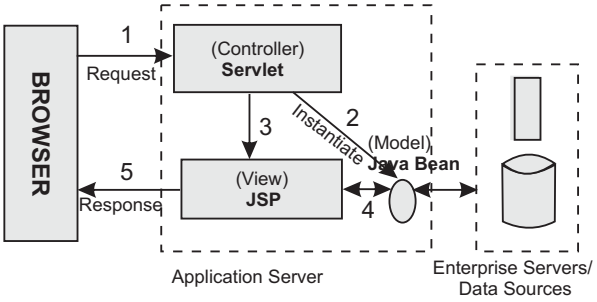

Fig. 1. JSP Model 2 architecture

The Model 2 (MVC) architecture is a hybrid approach for serving dynamic content, since it combines the use of both servlets and JSP. It takes advantage of the predominant strengths of both technologies, using JSP to generate the presentation layer and servlets to perform process-intensive tasks.

In the figure1, the servlet acts as the controller and is in charge of the request processing and the creation of any beans or objects used by the JSP, as well as deciding, depending on the user's actions, which JSP page to forward the request . Note particularly that there is no processing logic within the JSP page itself; it is simply responsible for retrieving any objects or beans that may have been previously created by the servlet, and extracting the dynamic content from that servlet for insertion within static templates. This approach typically results in the cleanest separation of presentation from content, leading to clear delineation of the roles and responsibilities of the developers and page designers on your programming team.

B. Struts2 Framework based on MVC2

Struts2 is an open-source web application framework being developed in Jakarta Project. Struts is also a set of cooperating classes, servlets, and JSP

tags that make up a reusable MVC 2 design. This definition implies that Struts is a framework, rather than a library, but Struts also contains an extensive tag library and utility classes that work independently of the framework. Struts uses the JSP/Servlet technology following the MVC model stated above. Struts combines Java servlets, Java Server Pages (JSP), custom tags, and message resources into a unified framework and saves the developer the time of coding an entire MVC model, a considerable task indeed.

The *Model* represent the enterprise information/data of the application. Anything that an application persists becomes a part of model. It also defines the manner of accessing such data and the business logic for data manipulation. In struts2 the model is implemented by the Action Component. It can be EJB, Java Data Objects (JDO) pattern.

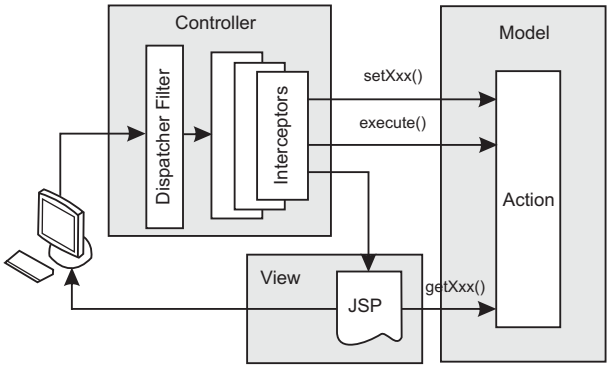

Fig. 2. Struts2 Framework Architecture

The *View* is simply a JSP file. There is no flow logic, no business logic, and no model information -- just tags. Tags are one of the things that make Struts unique compared to other frameworks like Velocity. Separate presentation logic from business and control logic.

All user interaction between the view and the model is managed by the *Controller*. All user requests to an MVC application flow through the controller. The controller intercepts such requests from View and passes it to the Model for appropriate action. The controller does not include any business logic. In struts2 the role of the controller is played by the Filter Dispatcher. This is a Servlet filter that examines each incoming request to determine the Action that will handle the request.

C. Spring Framework

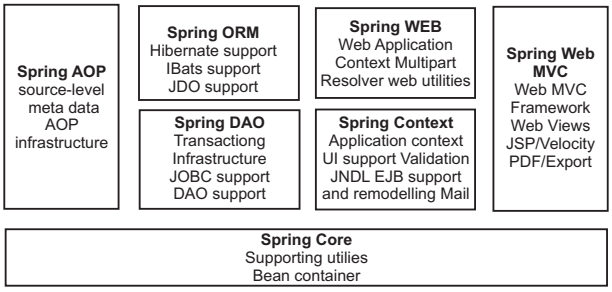

Fig. 3. Spring Framework Architecture

The Spring framework is a layered architecture Consisting of seven well-defined modules. Each module can exist individually or unite each other. The Spring modules are built on top of the core container, which defines how beans are created, configured, and managed, as shown in Figure 3. The main component of the core container is Bean Factory, which is an implementation of the Factory pattern. Bean Factory applies the Inversion of Control (IOC) pattern to separate an application's configuration and dependency specification from the actual application code.

This framework gives the programmers new patterns, so they can concentrate on the most important things in the project. Spring Framework can give us everything to build the corporate application and simultaneously gives us the possibility to use only some elements. We can differentiate one part that can quick integrate some other parts, this is IoC container. Besides IoC container there is another important component, that are the elements running inside the container called Beans. Beans are classes that are consistent with the JavaBeans specification. Any objects can be Beans in the Spring Framework. They don' t have to implement any interface and enlarge any class. The IoC container enables defining relations between Beans. Those relations are use when one Bean uses the other Bean to function right. IoC container is a pattern that is realized by dependency injection and placing the elements in the special container. That container can configure and match all dependences and objects before sharing it to the user. The IoC container shares two dependency injection methods:

- setter injection
- constructor injection

The main advantages of the Spring Framework are:

- integrating the existing solutions
- choice of suitable modules, only that which we need
- non invasion, the written code doesn't depend on the Spring Framework

In the questionnaire application system Spring Framework is an integrator between other system modules. His main role is to join other parts in one piece with use of Java Enterprise Edition, Hibernate and Struts.

D. Hibernate Framework Based on ORM

Working with both the object-oriented software and the relational database is a complicated task with Java Database Connectivity (JDBC) because there is mismatch between how data is represented in objects versus relational database. So with JDBC, developers have to write pure the Structured Query Language (SQL) statements to map an object model's data representation to a relational data model and its corresponding database schema.

Hibernate is a flexible and powerful Object-Relational Mapping (ORM) solution to map Java classes to database tables. It is a powerful, high performance object-relational persistence and query service. Hibernate allows developers to express queries in its own portable SQL extension (Hibernate Query Language (HQL), as well as in native SQL, or with an object-oriented criteria and example Application Programming Interface (API). Hibernate itself takes care of this mapping using XML files so developers don't need to write code for this.

Hibernate is an open source and it is free to use for both development and production deployments, which is a bridge between Java application and relational database and takes charge of mapping between Java objects and relational data. The inside of Hibernate packs the operation of accessing database by JDBC, which provides API of object-oriented database access to upper layer application. So developers can use the object programming thought to operate database sufficiently, caring for the bottom database structure unnecessarily.

Hibernate relieves the developer from 95 percent of common data persistence related programming

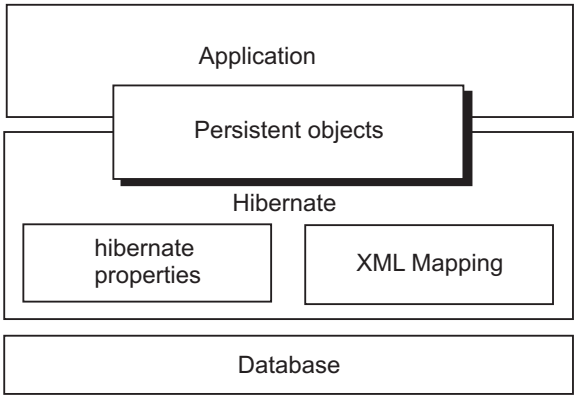

Fig. 4. Hibernate Architecture

tasks, compared to manual coding with SQL and the JDBC API. And it can integrate various Web server or application server, and nearly support all popular databases server.

IV. PROOF OF IMPLEMENTATION

This paper uses the example of a online bookstore dynamic E-commerce system to illustrate the design and implementation process. The system is divided into the following four functional modules: merchandise query, shopping management, order form management, after service, electron payment, administrators, user , merchandise, discount , content and advertisement management and so on.

In this paper, Bookstore management will be used as an example to tell how to realize by Struts2, Spring and Hibernate. This module is realized by three-tiered: Presentation tier, Middleware tier (Business Logic/Logic Tier) and Data tier as Figure 5.

A. Presentation Tier

The Bookshop manager can modify the property of the merchandise such as name, description, price, num and so on through the browser. The Jsp pages in presentation tier collect these data, and connect with the middle tier by the controller, and then the middle tier will connect with the data tier. At last, the data returned from the middle tier will be displayed by the JSP pages by using the Struts Taglibs.

B. Middleware Tier

This tier can be divide into four parts, that are web tier, service tier (business logic tier), DAO tier(Data Access Object) and PO tier as the followings:

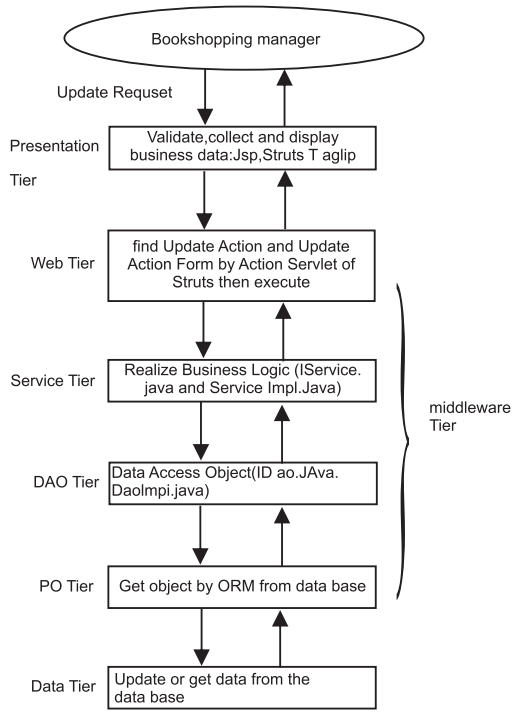

Fig. 5. Multi-tier Architecture for E-Commerce System

Web tier: the Controller of Struts is with responsibility

for the data exchange between presentation tier and business logic tier, and it transfers the business logic tier bean to deal with the merchandise information, then it transfers the returned data to the presentation tier to display.

Service tier (business logic tier): it realizes the business logic, and handles the merchandise information exchange between DAO component and Web tier.

DAO tier (Data Access Object): it encapsulates the functions of merchandise data query, modification, add and deletion. And it handles the merchandise information exchange between Service tier and PO tier.

*PO tier (persistent object tier).*the relational database table which stores merchandise information are mapped to data merchandise objects by the object/relational mapping tool of the Hibernate, then the developer can operate the database table of merchandise as the merchandise object.

C. Data Tier

In this paper, MSSQL server2000 is adopted as Database Servers in where information is stored and retrieved. This tier keeps data neutral and independent from application servers or business logic. Giving data its own tier also improves scalability and performance.

In this integrate pattern, whenever the user send the request from the interface to the server, the request will be sent to the ActionServlet of Struts, and then ActionServlet will send it to the Bean of Spring, in the case of which, the business objects can be highly configurable with the help of IOC Container. It means that now it is possible to use the Hibernate objects as Spring Beans to deal with the data between application and database, thus there is no need for the Application to depend on the low-level JDBC details like managing connection, dealing with statements and result sets. By Using Hibernate, the access of database becomes simply and object-oriented.

Struts2 framework is a classical implementation of MVC architecture. Hibernate is a powerful technology for persisting data, and it enables Application to access data from any database in a platform-independent manner. Spring is a dependency injection framework that supports IOC. The beauty of Spring is that it can integrate well with most of the prevailing popular technologies, thus integrate Struts, Spring and Hibernate is a very perfect pattern.

V. CONCLUSION

In this paper We developed the MVC2 architecture, which consists of concepts for web application middleware Struts2, Spring and Hibernate that supports custom and adaptable process . We discussed details of the architecture, including data distribution, decoupling through configuration. Our architecture addresses all requirements for web application middleware we previously identified. With our architecture, the distributed structure of a Web application can be defined and adapted through (re)configuration and without affecting its implementation. This gives developers more flexibility in constructing and customers more flexibility in deploying their web applications.

As proof-of-concept, we implemented a middleware Struts2, Spring and Hibernate framework based on the concepts of MVC2 architecture. The

middleware framework serves as an example of how the MVC2 architecture fit together and map to a concrete middleware implementation. We implemented a sample Web application on top of our framework and demonstrated that constructing custom process and then adapting them in several evolutionary steps later on is straightforward.

REFERENCES

[1] Mengjian Chen "A Dynamic E-commerce System Based on Middleware Technology". 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing.

[2] YANG Dezhi, ZHU Shifeng, ZHOU Shenglu, WANG Jiechen, CAI Anjuan. "The Design and Implement of Web MIS of Students Based on Servlet+JDBC" 2009 First International Workshop on Education Technology and Computer Science.

[3] R. Elmasri and S. Navathe. Fundamentals of Database Systems. 3rd Edition. Addison- Wesley, 2000.

[4] Model-view-controller,Accessed from: http: // www. roseindia.net/struts/struts2/

[5] Introduction to the Spring framework, Accessed from: http://www.ibm.com/developerworks/web/library/spring

[6] HIBERNATE - Relational Persistence for Idiomatic Java, Accessed from: http:// www.Hibernate.org/hib_docs/ v3/reference/en/html/preface.html

[7] I., Singh, B.Stearns, , M. Johnson, and E, Team, Designing Enterprise Applications with the J2EE Platform, Addison-Wesley, 2002.

K.Mani had done B.E(Computer Science and Engineering) in Christian College of Engineering and Technology, Oddanchatram-Dindigul. He is studying M.E (Computer Science and Engineering) in Ganadipathy Tulsi's Jain Engineering College, Vellore. He is worked as a Lecturer of the department of "Computer Science and Engineering" in VelTech MultiTech Engineering College, Avadi-Chennai.