A Decision Support System Based on SSH and DWR for the Retail Industry

Chunyang Wang and Bo Yuan
Division of Informatics, Graduate School at Shenzhen
Tsinghua University
Shenzhen 518055, P.R. China
tsinglong@163.com, yuanb@sz.tsinghua.edu.cn

Guowen Liu
Microprofit Electronics Co., Ltd.
Nanshan Technology Park
Shenzhen 518055, P.R. China
lgw@microprofit.com

***Abstract*—This paper presents the design of a retail decision support system with the B/S architecture. It is implemented under the SSH and DWR framework with AOP based access control following the principle of RBAC. We show that the design based on SSH and DWR is highly efficient and features good scalability, reusability and a responsive user interface.**

Keywords-SSH; DWR; RBAC; AOP; Decision Support

I. INTRODUCTION

With China's booming economy in the past decades, the retail industry in China has also experienced a strong growth. In particular, its outstanding performance has attracted world-wide attention during the global financial crisis. In the meantime, foreign investment is entering China's retail market in an unprecedented pace, and the competition becomes more and more intensive. In order to stay at a competitive position, enterprises in the retail industry must pay more attention to information technology, which can greatly enhance their competitiveness and provide them with strategic advantages [1].

For example, in a department store or a supermarket, customer flow information refers to the number of customers entering or staying at different sections of the store within in a fixed time period, which may change dramatically during the opening hours. A retail decision support system (RDSS) can analyze massive historic customer data to provide valuable information for daily decision support activities such as how to improve the effectiveness of staff resource scheduling and how to improve the shopping experience.

In general, the purpose of RDSS is to, given a potentially huge collection of customer information and sales data, conduct scientific decision-making activities as required and realize information sharing among various information systems. In many developed countries, RDSS has already been widely implemented and many enterprises in China have also adopted this technique to improve their competitiveness [2].

In this paper, we present the design of a RDSS for a major electronics store in China. The system architecture follows the principle of hierarchical design and the software layers are implemented using interface-oriented programming. The system employs SSH (Struts + Spring + Hibernate) to handle the page logic, business control and object persistence, and uses Struts to decouple the presentation layer from the business layer, and Hibernate to decouple the business layer from the persistence layer by DAO (Data Access Object) mode, respectively. A loose coupling between components is realized by Spring through IOC (Inversion of Control) and DWR (Direct Web Remoting) to improve the sensitivity of the response of user interface [3]. The system access control is realized in Spring by AOP (Aspect Oriented Programming). With the above design strategy and selected techniques, the system is expected to have good security, sensitivity, scalability as well as reusability.

In the rest part of this paper, Section II describes the overall system framework based on SSH and DWR. The access control strategy and system functions are specified in Section III and Section IV, respectively. The system implementation details are presented in Section V and this paper is concluded in Section VI with some analysis and directions for future work.

II. SYSTEM FRAMEWORK

To facilitate the design, implementation and maintenance of RDSS, the system follows the principle of hierarchical design. According to the system functions, different operations are grouped and assigned to different layers, avoiding unnecessary association among the layers as much as possible. Interfaces are used to communicate between two layers. The lower layer provides invoked interface for the upper layer and its details of realization are transparent to the upper one, thereby reducing the coupling of the system and resulting in good scalability, reusability and maintainability [3].

According to the design principle, the system combines SSH [8-11] based on the MVC (Model-View-Controller) mode with DWR [12-15] based on Ajax (Asynchronous JavaScript and XML) technology. The framework of the proposed system is shown in Figure 1.

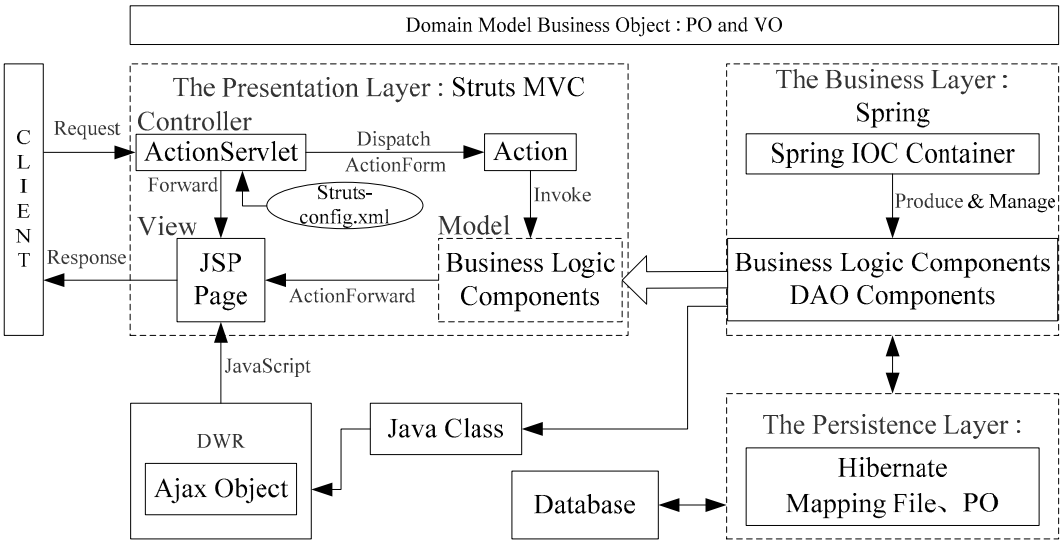

Figure 1. Framework of the Decision Support System.

A. *The Presentation Layer*

The presentation layer based on Struts is responsible for handling system's page display, responding to user requests, calling the lower layer's business logic components, and returning the results to the client. Struts has the advantages of being modular, reusable and flexible. It achieves the MVC pattern by Servlet and JSP technology, and decouples the presentation layer from the business layer by Controller realizing a layered structure.

B. *The Business Layer*

The business layer based on Spring is the system's core layer. It is responsible for receiving requests from the presentation layer, executing the calls to the persistence layer, performing application logic and transaction processing etc. Spring organizes various objects together in a loosely coupled manner through its core mechanism called IOC or DI (Dependency Injection), and the calling of the objects is interface-oriented. In the business layer, IOC is responsible for providing business model components and collaboration objects (DAO components) for Action to complete business logic, and it supplys buffer pool, transaction processing and so on to improve system's performance and ensure data integrity.

C. *The Persistence Layer*

The persistence layer based on Hibernate is responsible for data access, database backup and synchronization and so on. Hibernate packages access details of database by DAO, and all access to database is completed by calling the DAO objects, which generally include the basic CRUD (Create, Retrieve, Update and Delete) operations to PO (Persistent Object). Hibernate associates PO with database tables by its object-relational mapping file (*. Hbm.xml) and creates a one-to-one correspondence between attributes in PO and fields of the database table. By operating PO to conduct the basic CRUD operations on the database tables, the complexity of data access can be simplified.

D. *DWR*

Although SSH has many advantages, there is a lack of sensitivity on the response to user interface. SSH adopts a synchronous "request → response" mode: the application sends user request to server and sits idle, waiting until the new page returns from the server. Page refreshing depends on reloading page from server completely, leading to a low sensitivity on the response to user interface. To solve this issue of SSH, the system uses Ajax, adding an Ajax engine between client and server. It allows client and server to interact in an asynchronous manner, so that users can continue other work without waiting for the response from server.

After the request is finished on the server side, results are submitted to Ajax engine. Ajax engine uses JavaScript to update the corresponding user interface instead of refreshing the entire page to give users a feeling of real-time response. Therefore, the presentation layer of SSH is joined with an open source Ajax framework DWR. DWR configures the service (Java class) that is public to client in dwr.xml and dynamically generates JavaScript objects by Java class so that the page is able to use these objects to call the service, which makes the system more dynamic and responsive.

III. ACCESS CONTROL

Traditionally, the system administrator grants permissions to users directly to implement access control. The relationship between users and permissions is many-to-many and users can access various system modules under their own set of permissions. However, when the type and number of user increase, this strategy cannot reflect the

relationship among user groups effectively, which is not convenient for user management and may easily result in information leakage [4].

RBAC (Role-Based Access Control) adds role between users and permissions, assigns permissions to role, and grants users access permissions by giving users different roles. RBAC simplifies the access control model and avoids possible defects in the traditional access control strategy by combining grades and constraints to achieve a variety of security policies [5].

The system implements permission control as a separate module by AOP. AOP configures the permission control module in Spring as an aspect, responsible for verification by dynamically executing the permission control module when the core business is called. Therefore, it has the advantages of low coupling, easy expansion and strong versatility. In the meantime, AOP can add functionality dynamically without modifying the source code through pre-compilation and dynamic proxy in runtime.

IV. System Functions

A. *Structure*

The RDSS contains customer counting subsystem, data query subsystem, data analysis subsystem, anomaly alarm subsystem and system setting subsystem (Figures 2-6).

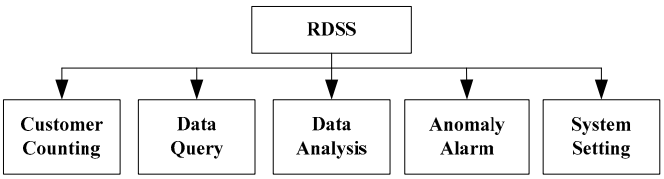

Figure 2. The overall structure of RDSS.

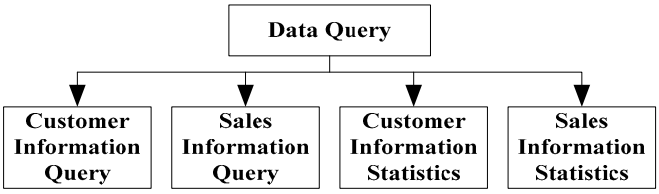

Figure 3. The structure of the data query subsystem.

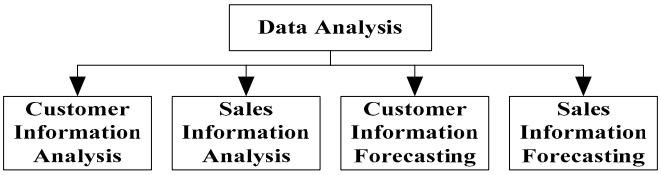

Figure 4. The structure of the data analysis subsystem.

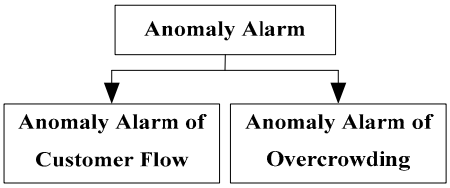

Figure 5. The structure of the anomaly alarm subsystem.

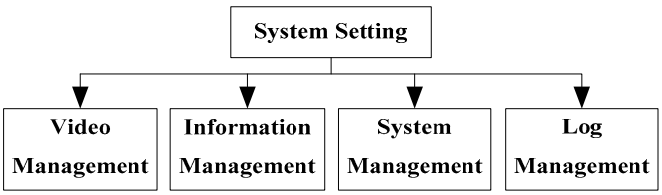

Figure 6. The structure of the system setting subsystem.

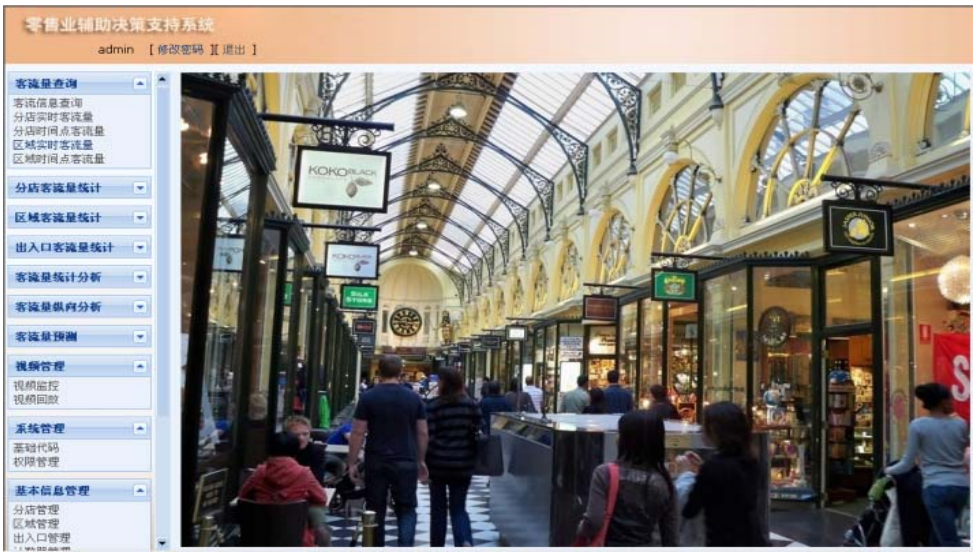

Figure 7. An example of the system interface.

Figure 8. The query interface for real-time customer flow.

B. *Function Description*

The RDSS realizes a smooth integration of customer and sales information. It provides comprehensive and effective functions including query, analysis and forecasting of customer and sales information, as well as customer anomaly alarm. It displays valuable information via charts and serves as a reliable basis for management decision-making. The system interface is shown in Figure 7.

The customer counting subsystem is based on an intelligent technology for tracking and recognizing moving targets. It incorporates SVM (support vector machine), key feature matching techniques and statistical models. It transforms customer video images into a data flow of inbound and outbound customers through motion analysis and feature classification. The real-time video images (head and shoulders images) of customers are collected by a camera installed over the entrance to the store or a specific region of store.

The data query subsystem provides statistical information and supports various queries of customer and sales information. Users can not only query real-time information but also access customer and sales information based on time, activities, weather etc. The query interface of the customer flow module is shown in Figure 8.

The data analysis subsystem provides the analysis and forecasting function for customer and sales information. Through the data analysis system, users can not only analyze customer information and sales data according to the amount of sales, purchase frequency and time, per capita staying time, per capita consumption, etc. but also forecast the customer and sales information in the future. Users can timely and accurately grasp the trend of customer and sales information and prepare in advance to deal with the changes of customer and sales.

The anomaly alarm subsystem is a critical service for ensuring the normal operation of a store. It includes the detection of abnormal customer flow and abnormal number of customers. For example, when the number of customers staying in the store or a region of store is more than a threshold value, the anomaly alarm subsystem will raise the alarm and notify the shop staff in charge who can then follow appropriate strategies to ensure the safe and orderly running of store based on the alarm information.

The system setting subsystem includes video management, information management, system management and log management. Video management includes video surveillance and video playback. Information management sets up and manages some basic information, including store management, region of store management, import and export management, counter management, and so on. System management includes permissions management, password changing and others, which is responsible for the authentication of users as well as controlling the operation permissions of user.

V. SYSTEM IMPLEMENTATION

Among the many modules in RDSS, we choose the permission management subsystem as an example to explain the design based on SSH and DWR.

System development environment: MyEclipse 6.0.1, JDK 1.6.0_04, Microsoft SQL Server 2005, Tomcat 6.0.18, Struts 2.0.11.2, Spring 2.0.1, Hibernate 3, DWR 2.0.5.

System development procedure: VO → PO → DAO → SERVICE → ACTION → SPRING → JSP.

The system design should first determine the business entities, and then create and implement the domain model through the analysis of business entities, establishing a Java object (PO) of the persistence layer for each entity.

A. The Presentation Layer

Hibernate creates a mapping file (*.Hbm.xml) for each PO and associates PO with database by mapping file. Hibernate also designs the interface and implementation of DAO based on PO. Spring allows DAO in Hibernate to inherit tool class (HibernateDaoSupport) in Spring to provide good support for DAO so the that functions provided by tool class can be used to access database.

B. The Business Layer

Spring implements business logic and functions as a core framework, responsible for integrating Struts, Spring, Hibernate and DWR and generating related objects according to the configuration file. Spring creates the interface and implementation of service objects by business modelling and connects components of the layers in a loosely coupled manner. Through the configuration file applicationContext.xml, Spring connects the interface and implementation defined by the business layer with those defined by DAO and provides essential transaction management for them.

C. The Persistence Layer

For Struts, the persistence layer employs JSP and TagLib library of Struts to handle the display, and the request (*.do) is mapped to the corresponding Action through ActionServlet. The persistence layer adopts Action to call the service components of business logic managed by Spring and jumps to the response page specified by the Forward object according to the processing results. Achieving this functionality involves three major aspects: configuring the mapping from specified request path to the corresponding Action class in struts.xml, integrating Struts and Spring, and the implementation class of the Action of users, which is in charge of actual processing.

DWR is used to improve the responsiveness of user interface. DWR can avoid the jump of pages and increase the usability of the JSP pages in the presentation layer. The major DWR operations are as follows. Firstly, adding the mapping of DwrServlet in web.xml to load DWR framework. Secondly, configuring DWR and adding dwr.xml. The create element tells DWR to make publicly available the server-side class (beanName specified by param) requested by Ajax and makes DWR generate the corresponding javascript library for these beanName in the runtime so that the JSP page can use the library to call Java classes directly.

VI. CONCLUSION

This paper presents a RDSS based on SSH and DWR, which follows the principle of hierarchical design. The software layers are implemented using interface-oriented programming. The system uses Struts to decouple the presentation layer from the business layer by the MVC mode, and uses Hibernate to decouple the business layer from the persistence layer by the DAO mode. The object-creating and relationship-injecting of the layers are achieved at runtime to realize a loose coupling between components so that the system can have good scalability, reusability and maintainability. In addition, the responsiveness of user interface is improved by DWR.

The RDSS described above is effective and capable of handling various daily activities. In the future, we will further enhance its core decision support function by employing advanced data mining techniques to analyse both historical and real-time customer data and sales information and optimizing the software architecture to make it more effective in handling massive amounts of data.

REFERENCES

[1] H. Y. Wang, "The status and trends research of China retail informationization," Journal of Liaoning Administration College, vol. 12, pp. 88-89, 2010.

[2] Y. M. Wang, "Customer analysis system adding momentum for commercial running," China Computer & Communication, pp. 36-38, 2006.

[3] Y. Chen, S. Li and Y. N. Zhao, "Research and application of Web framework based on SSH + DWR," Journal of Nanjing University of Information Science and Technology: Natural Science Edition, vol. 2, pp. 455-460, 2010.

[4] W. L. Guo, H. J. Jiang and S. G. Liu, "Based on SSH framework RBAC design and Implementation," Computer Engineering & Software, vol. 32, pp. 47-48, 2011.

[5] B. Yu and W. Shen, "Design of college student service information management platform based on SSH," Modern Electronics Technique, vol. 35, pp. 47-49, 2012.

[6] Y. Y. Hao, L. Sun and R. Y. Chen, "Design and application of RBAC system based on AOP," Network & Computer Security, pp. 43-45, 2009.

[7] Z. X. Gao and Z. X. Li, "Role based access control and implementation under web environment," Computer Engineering, vol. 30, pp. 133-135, 2007.

[8] F. Q. Wang, Unveil Spring, Beijing: Posts & Telecom Press, 2009.

[9] G. Li, Elaboration of application development based on the integration of Struts + Hibernate + Spring, Beijing: Tsinghua University Press, 2007.

[10] Y. C. Ren, T. Xing, Z. F. Xing and J. C. Zheng, "Application research for integrated SSH combination framework to achieve MVC mode," Computational and Information Sciences, Chengdu, 2011, pp. 499-502.

[11] F. Liu, H. X. Guo and B. Fu, "The research of web application framework based on SSH," Business and Information Management, Wuhan, 2008, pp. 169-172.

[12] N. H. Wang and X. Jin, "The Ajax techniques and DWR frame realization," Techniques of Automation & Application, vol. 26, pp. 92-94, 2007.

[13] C. P. Xu, M. Zhang and J. Zhang, "Ajax-based J2EE security application framework," Computer Engineering, vol. 36, pp. 110-111, 2010.

[14] H. Li, H. W. Zhang, Q. Yang, G. Z. Zhang and B. G. Wei, "Research and implementation of an booking system for laboratory based on SSH and Ajax framework technology," Journal of Gansu Lianhe University: Natural Science Edition, vol. 25, pp. 79-81, 2011.

[15] L. Zhang, F. Y. Zhang and S. J. Wei, "Web application design and realization based on DWR frame," Computer Technology and Development, vol. 18, pp. 84-87, 2008.