Data Expression Methods in Enterprise application using J2EE Architecture

Zhang Xiaoshuan[1,2],Chen Peijun[1], Zhao Ming[1]
[1]College of Information and Electrical Engineering
China Agricultural University
Qinghua Donglu 17, Haidian, Bejing
P. R China, 100083
[2]Jiangsu Provincial Key Laboratory of Computer Information Processing Technology
Suzhou University
Shixinjie 1, Suzhou, Jiangsu
P. R China, 215006
Corresponding author: zhaoming@cau.edu.cn

Abstract: - With use of the Internet, web browser-enabled software applications are developed rapidly. Nowadays enterprises are paying more attention than ever on developing their business on the Web. The demand for rapidly delivering high quality web enabled software applications to the end users is high. The enterprise software developers are making efforts to develop the enterprise software applications that not only satisfy the business needs but also achieve the high quality within a short development process. However, the enterprise software development is normally complicated. For example, factors like the different enterprise environment, the critical business demand, the lack of knowledge and experience from the developers and so on can make the enterprise software development become rather hard work. Under such circumstances, something is needed to simplify the enterprise software development and also enable the sharing of the knowledge and experiences among the different developers. J2EE architecture is becoming the mainstream framework to develop the more complex enterprise application. The important problem arises, which is how to manage to each layer in J2EE and control the interior business data and the customer data, given that J2EE architecture has many complicated layers. There is respective data presentation in relevant layer of J2EE, with strictly rules to access and transform data objects between these layers. This paper adopts the case study method to discuss the data expression of J2EE and the detailed data expressing methods in corresponding layer with some instances. According to the Façade pattern of J2EE, it should control the access to data persistence layer seriously to protect important enterprise data and avoid showing database pattern to client directly. The data persistence layer hides behind the business logic layer and provide business interfaces via the business logic layer to communicate with the presentation layer to package the data persistence layer, separates the data persistence layer from the presentation layer.

Key-Words: - Data Expression, Persistence Layer, Façade, ORM, e-commerce

1 Introduction

With use of the Internet, web browser-enabled software applications are developed rapidly. Nowadays enterprises are paying more attention than ever on developing their business on the Web. The demand for rapidly delivering high quality web enabled software applications to the end users is high. The enterprise software developers are making efforts to develop the enterprise software applications that not only satisfy the business needs but also achieve the high quality within a short development process. However, the enterprise software development is normally complicated. For example, factors like the different enterprise environment, the critical business demand, the lack of knowledge and experience from the developers and so on can make the enterprise software development become rather hard work. Under such circumstances, something is needed to simplify the enterprise software development and also enable the sharing of the knowledge and experiences among the different developers.

J2EE architecture is becoming the mainstream framework to develop the more complex enterprise application. The important problem arises, which is how to manage to each layer in J2EE and control the interior business data and the customer data, given that J2EE architecture has many complicated layers. There is respective data presentation in relevant layer

of J2EE, with strictly rules to access and transform data objects between these layers.

Whether they're internal applications for employee productivity, or Internet applications for specialized customer or vendor services, (jave.sun.com), portability and scalability are also important for long term viability. Enterprise applications must scale from small working prototypes and test cases to complete 24 x 7, enterprise-wide services, accessible by tens, hundreds, or even thousands of clients simultaneously.

For enabling services that allow multiple processes running on one or more machines to interact across a Internet network, middleware technology, one kind of the dependence to the server end technology, evolves to provide for interoperability in support of the move to coherent distributed architectures, which are used most often to support and simplify complex, distributed applications is also increasing[1].

So the enterprise application should be related with those middleware which are flexible and can be transplanted when the Enterprise try to develop and apply the enterprise application given that the application should has the ability of dealing concurrently with tens of thousands of users uninterruptedly all day long.

However, multitier applications are hard to architect. They require bringing together a variety of skill sets and resources, legacy data and legacy code. In today's heterogeneous environment, enterprise applications have to integrate services from a variety of vendors with a diverse set of application models and other standards. Industry experience shows that integrating these resources can take up to 50% of application development time.

Apparently the internet-based multitier applications application has more complex than the traditional information system. How to design, develop and maintain the more complex e-commerce website are becoming the key problems when the enterprise managers make the decision of information system.

There have mainstream solution provide by Microsoft and SUN Company. Comapred with .NET framework by Microsoft, the Java 2 Platform provided by SUN, Enterprise Edition (J2EE) defines the standard for developing multitier enterprise applications. The J2EE platform simplifies enterprise applications by basing them on standardized, modular components, by providing a complete set of services to those components, and by handling many details of application behavior automatically, without complex programming.

So adoption of J2EE can not only simplify the foundation of the enterprise level application program, but also cause the designers and the programmers to distribute the function in each discreteness of the server end [2, 3].

Once J2EE is adopted as the technology framework for developing e-commerce application at large-scale enterprise level, the important problem arises, which is how to manage to each layer in J2EE and control the interior business data and the customer data given that J2EE adopts the layered thought, and subdivides the process of the corresponding response or the resources that obtains from the user into multi-layer to deal with, and each layer is realized by the component technology. It needs to coordinate the relations of each layer well, and increases the cohesion of each layer, but maintains the incompact coupling.

There is respective data presentation in relevant layer of J2EE, with strictly rules to access and transform data objects between these layers. This paper discusses the data expression in e-commerce application at enterprise level.

The following sections are organized as the followings: section 2 describes the basic material and methods, section 3 gives the research and experiential results, last section concludes the works.

2 Material and methods

2.1 The multi-layered architecture and the data expression methods of J2EE

The Java 2 Platform, Enterprise Edition (J2EE) defines the standard for developing multitier enterprise applications. The J2EE platform simplifies enterprise applications by basing them on standardized, modular components, by providing a complete set of services to those components, and by handling many details of application behavior automatically, without complex programming.

The multi-layered architecture of J2EE [3] mainly includes the client layer (the behave layer), the presentation layer (the Web layer), the business logic layer (the application layer) and the data persistence layer (the EIS layer). Each layer has its specific function. Through providing components to the application program, J2EE realizes the multi-layered architecture function [4].

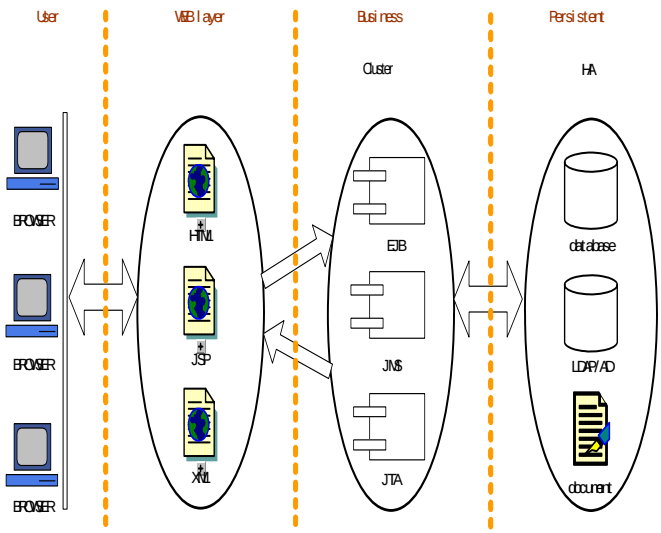

Fig 1 J2EE Architecture system

J2EE multi-layer distributed applications using model for the application of logic and functional components of the various application components according to their host machines located in different layers. In fact, the sun J2EE design in the first place was to address the two-layer model (client/server) defects in the traditional model, the client played a role is over bloated, in this model, the first deployment of the relatively easy, but it is difficult to upgrade or improve, which can reach sexual ideal. often based on a proprietary database of the agreement-usually some agreement. Makes it very difficult to reuse business logic and interface logic. Now the multi-enterprise-level J2EE application model to a two-layer model is divided into many layers of all different levels. A multi-level applications can provide an independent service for each of the different layers.

These four levels are running on the client machine client layer (Client layer), the operation of the Web server A Web (Web layer), EJB server operations in the business (the Business layer) and enterprise information server operating system layer in the EIS (Enterprise Information System layer), which the Web layer and operational levels in the middle to form a three-layer J2EE application layer is a two-layer or the client layer and storage layer enterprise information system. Under normal circumstances, many operators open EJB server and Web server combined product releases, known as application server or J2EE server.

In the J2EE architecture, each level has its respective data expression method:

(1) The data expression of the Web layer is FormBean [5] , and the data root in the HTML Form POST;

(2) The data expression of the business logic layer is the VO (Value Object);

(3) The data expression of the persistence layer is the PO (Persistence Object), and the data root in the database.

The following is explaining the function of each layer.

2.1.1 The client layer

According J2EE multi-layer architecture, the model is composed of four layers, which includes client layer, Web layer, EJB layer and data layer.

J2EE supports many kinds of client types [6]. Because of the B/S structure [7], the main body of the program runs at the server end, and the client server does not need to undertake the complex calculate duty (this is the so-called " the thin client server "). The major function of the client layer is to enable the users to carry on the communication smoothly with the server through the user interface.

Client is a browser application procedures and client layer components, Client level components that can be as a Web server on the Web browser, and can also be based on traditional methods (Web-based) that is independent of the application procedures can be completed Thin-client/Server impossible task.

There are three kinds of technologies on the client layer: HTML (a general electronic commerce network, and is also the most universal kind), the radio equipment (can get information anytime and anywhere), and the Applet or the Java application program (can be used to realize the complex and fast user interface) and so on.

2.1.2 The presentation layer

The presentation layer is also called the Web layer [8], it runs in the J2EE Web vessel. The main responsibility of the presentation layer is to present the car service reservation business information. As mentioned earlier, the business information must be rendered in a flexible and visualized way. The basic layout of the presentation layer is being constructed as a web calendar, which visualizes the real business schedules. With the additional associated information, thefinal users can quickly observe not only the reservations and schedule but also the supplementary information related to the schedule, reservation and even the employee's workload.

Other responsibilities at the presentation layer include the session data management, client access control, view rendering and configuration, and validation.

When working with the business layer, the presentation layer processes the user’s request into a particular command and passes the command to the

business layer. In the meantime, the presentation layer redirects the user to the result view where the result data responded from the business layer can be captured. One thing to be mentioned here is that a controller layer resides between presentation and application logic layer is used to manage the access control and distribution of different requests throughout the whole application.

Ever since Tim Berners-Lee created the first Web-based system, presentation layers for n-tiered systems have undergone a revolution. Before that time, developers were forced to create client systems tightly coupled with their corresponding servers. Using only the basic HTTP protocol, a Web server, and HTML, developers were able to deliver document-based applications to users, regardless of what hardware or software platform they used. This approach had some fundamental problems for application developers: while HTML succeeded at delivering document-based data, it proved unsuitable for creating rich applications—applications with ample real-time feedback for users.

To address these shortcomings, developers began exploiting some of the features available in modern (post Netscape Navigator 2.0) Web browsers, namely Java and JavaScript. For the first time, developers were able to leverage the Web browser platform as a delivery mechanism for rich, platform-neutral applications. The actual user experience of using applets, however, never really matched the expectations. Applets required users to already have the Java Runtime Environment (JRE) installed and a Java Plug-in for their Web browser. In addition to the steps needed to set up a client system to execute Java applets, the client machine needed to download the actual Java applet. This could often prove time consuming, especially over slower Internet connections.

Now in the J2EE architecture's presentation tier, servlets and JSPs generally render data derived from the business logic tier as HTML for browser presentation. Implementing your application in HTML offers many advantages: HTML is easy to write, very lightweight, and looks aesthetic. Its major function is to deal with the HTTP request, and produce the response of HTTP dynamically according to the Servlet and JSP in the Web server.

2.1.3 The business logic layer

A Business logic layer (BLL) is a software engineering practice of compartmentalizing the rules and calculations of a software application from its other design elements. This Business logic layer is usually one of the tiers in a multitier architecture. It separates the business logic from other modules, such as the Data access layer and User interface.

By doing this, the business logic of an application can often withstand modifications or replacements of other tiers. For example, in an application with a properly separated Business logic layer and Data access layer, the Data access layer could be rewritten to retrieve data from a different database, without affecting any of the business logic. This practice allows software application development to be more effectively split into teams, with each team working on a different tier simultaneously.

Within a BLL objects can further be partitioned into Business Processes (aka Business Activities) and Business Entities. Business Process objects typically implement the Controller Pattern, ie: they contain no data elements but have methods that orchestrate interaction among Business Entities.

The different technologies the business logic layer [9] use run in the corresponding vessels. They are mainly used to realize the business logic in the enterprise application program. The so-called business logic refers to the processing demand to the data carried by the specific enterprise (client). For example, in the telecommunication enterprise, there is a series of operations such as inquiring the phone bill, filling the account, calculating the cost and so on. Each enterprise has its unique business logic because of there own distinctive quality, so it needs to be realized in the business logic layer.

In the actual project development, the business logic lay is very important; it provides concurrency, flexibility, life cycle management, and fault tolerance and so on for the entire project.

2.1.4 The persistence layer

The persistence layer [10] is a core problem of data expression, the data expression of other layers is all working for the persistence layer in fact. The persistence layer describes the business data of the application program, and simulates the real entity and the business flow of the organizations, so it is the foundation of the enterprise application.

A persistence layer encapsulates the behavior needed to make objects persistent, in other words to read, write, and delete objects to/from permanent storage.

By conforming to this class-type architecture the robustness of your source code increases dramatically due to reduced coupling within your application. For the user-interface layer to obtain information it must interact with objects in the domain/business layer, which in turn interact with the persistence layer to obtain the objects stored in your persistence mechanisms. This is an important feature

of the class-type architecture-by not allowing the user interface of your application to directly access information stored in your persistence mechanism you effectively de-couple the user interface from the persistence schema. The implication is that you are now in a position to change the way that objects are stored, perhaps you want to reorganize the tables of a relational database or port from the persistence mechanism of one vendor to that of another, without having to rewrite your screens and reports.

The business object is the simple software abstract of the real world entity, it represents some concrete object in the business domain. In the project, as the BO has provided a whole set common term and thought that can express the real world very directly, thus sharing information in the enterprise will be unblocked by using it.

In the enterprise development, it always needs to persistent the BO and make the data keep long haul. This is the function of the persistence layer.

2.2 The selected case description

Some electricity generation management Ltd. has many subordinate power plants. Presently, the local area networks of the group headquarters and the subsidiary company respective have been found, and some of the subsidiary companies have already implemented their management information system (MIS). But because of not carrying on the comprehensive data collection through the software and the network construction, the headquarters is not able to grasp the management information of the subsidiary company by observing the data. As a result, there has formed a information isolated island between the subsidiary company and the headquarters.

The construction of this project can directly obtain the data of the production management, the business management, and the customer relations and so on through the VPN from the MIS and the system-related of its subordinate power plants. This will provide a reliable network foundation for establishing the power plant appraisal system, making development programming, and forecasting the business trade.

The functional module division of the entire application system including evaluation module, data acquisition module, data analysis module, user management module etc. By realizing this application system, it will construct an assistant decision support system which can across network and the system as well as a data analysis platform faced to the management decision.

3 Results and discussion

3.1 data expression methods in J2EE architecture system

In the J2EE multi-layered architecture, it should strictly control the visit to the persistence layer. One of the reasons that J2EE adopts the layered thought is to protect the important business data, and avoid exposing the database pattern directly to the client server. Generally, the presentation layer interacts with the database by the business layer and the persistence layer, as the Figure 2 shows.

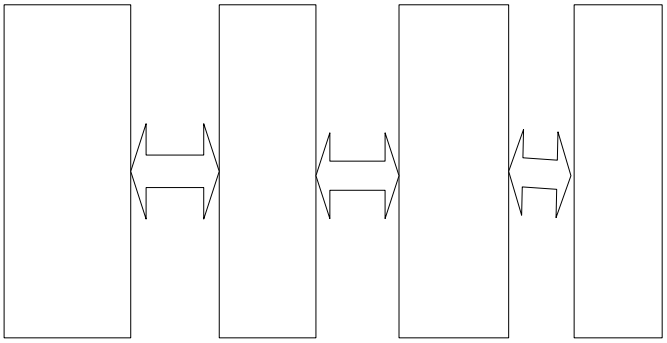

Fig.2 the interaction between the layers

The word Façade in English means the frontage of the building. But in the design pattern, it means to hide the facility behind the interface (the positive pattern Façade Pattern) by the uniform and simple interface. According to this thought that hides the persistence layer behind the business logic layer, and connects with the presentation layer through the interface that the business layer provided, to realize the separation between the persistence layer and the presentation layer.

In fact, in the actual project development, even if using the Façade Pattern, , it is possible to expose the presentation layer to the presentation layer even the client server so long as you do not pay attention slightly. Take some Electrical Power Office as example; it needs to inquire the sale situation of its substation. Obviously, the presentation layer is used to show the sales information, and the business logic layer carries on the corresponding transfer (the Entity Bean or some other ORM frame package object) to the persistence layer according to the request which has been passed by the presentation layer. After obtaining the corresponding business object of the persistence layer, it transmits this BO to the presentation layer, and afterwards the presentation layer transfers the geter method of BO, and shows the data or the information to the users.

It looks nature to do this on the surface. However, when transfer the BO back to the presentation layer, middle, although it has passed through the

intermediate layer (the business layer), in fact it makes the presentation layer contact with the persistence layer directly. Because of the frequent read-write to the database during the persistence layer operation, it must use affair inevitably.

But every time the vessel starts an affair, it will read data first from the database and arrive at the presentation layer after the multi-layered transmission. This kind of extra round-trip operation in the database will aggravate the server's expenses enormously. Therefore generally speaking, it should make the business layer approach to the persistence layer near and be far away from the presentation layer, all the affairs should be put in the business layer to carry on the processing(Fig 3)

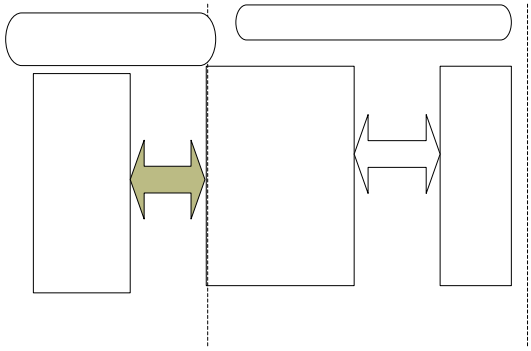

Fig 3 the transaction control

Value Object (VO) is the best method to resolve the problem how the business data in persistent layer can be called from business layer and submitted to presentation layer while the persistent layer avoid to directly connect with the presentation layer. The data business layer acquire from persistent layer are not directly transmit to the presentation layer but evaluate to the VO. Then the presentation layer read the information from VO. The so-called value object is a JaveBean, which encapsulates the business data the presentation required (see Fig 4). At the result, the presentation layer can read the required information directly from the VO, not read the data. It accords with the thought of frontal model: business layer encapsulates persistent layer.

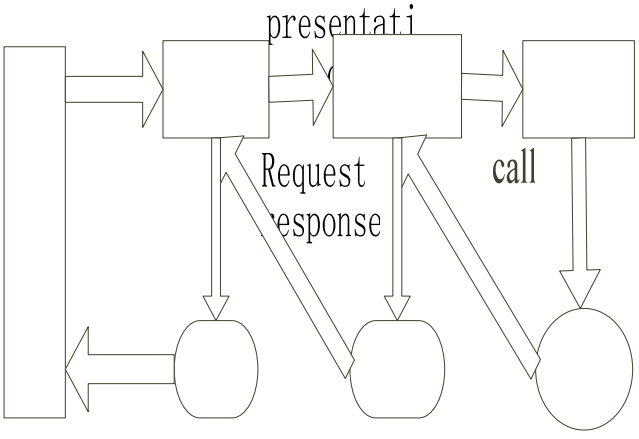

Fig 4 the date call and return based on VO

3.2 The overall framework and date flow structure for the test system

To realize the above functions, it first needs to choose a kind of appropriate architecture [11]. After overall evaluation on each kind of platform and the architecture system based on this platform, it chose J2EE finally.

The application architecture divides into three tiers in the figure: the data tier, the application tier, and the channels tier. And the application tier is subdivided to three parts: the application server, the framework, and the application(Fig 5).

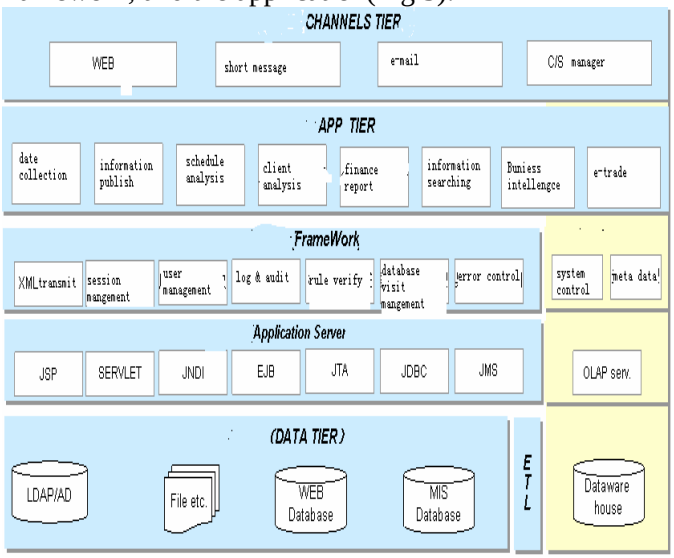

Fig 5 the overall framework for the system

With the application of EAI, the website may realize the application integration with other carry-over systems. With the ETL tool, it could realize the business intelligence (BI) application of the application data. The system development adopts the Model/View/Controller as the basic framework.(Fig 6).

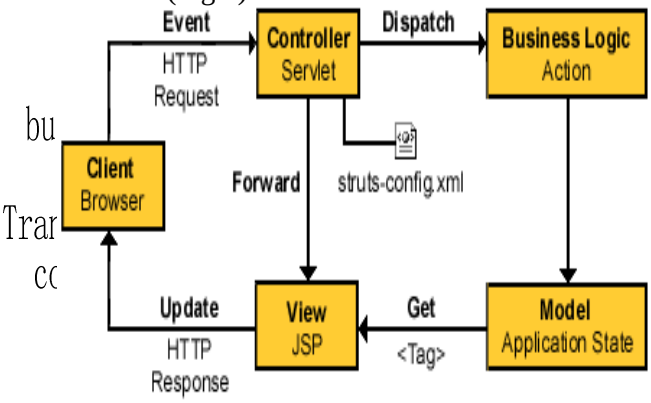

Fig6 the MVC model

During the MVC mode realization, the JSP was adopted to develop the version V in the MVC mode, the controller was developed employed the Struts

framework. The open resource Hibernate is used as the scheme of the data persistent layer. The MVC mode has some values: it can separate the version, data and business logic and decrease the minimum interrelationship among them.

Hibernate is a powerful, high performance object/relational persistence and query service. Hibernate lets the system develop engineering develop persistent classes following object-oriented idiom - including association, inheritance, polymorphism, composition, and collections. Hibernate allows you to express queries in its own portable SQL extension (HQL), as well as in native SQL, or with an object-oriented Criteria and Example API (http://www.hibernate.org/).

Unlike many other persistence solutions, Hibernate does not hide the power of SQL from you and guarantees that your investment in relational technology and knowledge is as valid as always.

3.3 The J2EE server setup and realization

Before using the Struts, the JSP web server must be prepared so that the web server can deal with and transmit the request to relevant controller—Struts ActionServlet. The configure information should be read when the web server start up。

The following is relevant code resource.

3.3.1 Web server configure

The following is basic web server configure information.

Web..XML

```
<web-app>

<display-name>power</display-na
me>
  <filter>
    <filter-name>auth</filter-name>

<filter-class>XXXX.SignonFilter</
filter-class>
  </filter>
  ……
  <filter-mapping>// class mapping
    <filter-name> SignonFilter
</filter-name>/
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  ……
</web-app>
```

3.3.2 Struts-config.xml

The configure document of struts-config.xml is the core of the framework. The web.xml document only defines where a request transmits. Struts-config.xml manages and control all of the mapping relationship between require and action.

```
<struts-config>
  <form-beans>
<form-bean
name="userCheckActionForm"
type="XXXX.logon.UserCheckAct
ionForm" />
……//formbean logic name and
physical location.
  </form-beans>
  <action-mappings>//Lists all of
mapping between require and action
    <action path="/sczhrb_qjAction"
type="XXXX.action.Sczhrb_qjActi
on">//Define the URL mapping
      <forward name="success"
path="/success.jsp" />//the
connection webpage after the
transaction success
    </action>
          ……
  </action-mappings>
  <message-resources
parameter="XXXX.ApplicationRes
ources_zh" />//define the resource
document
</struts-config>
```

3.3.3 Filter realization

The filter obtains the user information from the session. If the user is not authorized, the webpage will redirect the logon webpage. In addition, the redirected webpage can redefine based on the requirements.

```
Object id = null;
try{

id=session.getAttribute("userInfoB
ean");//get the user information
from the session

if(id!=null){chain.doFilter(req,res);
}//if the user information in the
session is not null, continue the next
to filter
   else{response.sendRedirect(LOG
IN_PAGE);}//if the user is not
authorized, the webpage will be
redirected the logon webpage
```

```
}catch(Exception e){ //error capture
   System.out.println(e.toString());//
error output
}
```

3.4 data expression methods realization at the e-commerce website at enterprise level

Regarding to the design of the data persistence layer, although there are many kinds of data memorizer, in the business data aspect, the DBMS on the database server basically uses the relational database in the current major enterprise developments. How to map the object and the relational database, and enhance the system capability, is the content needs to be considerate when design the data persistence layer.

The data expression in the business logic layer is VO, and the presentation layer is Formbean. The former in fact is a JavaBean, and the latter can be realized by inheriting the Formbean which is a class in the Struts. The Formbean class is used to receive the user data, and validate the grammar and others of the data by using the Validate () method. Moreover, it also extracts data from the business layer and gives them back to the users.

Generally speaking, the FormBean is the data expression in web layer and can't be transmitted to business layer. PO is the data expression of persistent layer. In special context, for example Hibernate, PO can be in business layer instead of VO. But both of VO and PO must limit in the business layer, the maximum scope is arriving the control of web layer, not scattered into view. The data transmission between Formbean and PO must be executed in the action.

The followings are parts of algorithms of code resource for realizing data expression.

3.4.1 Data expression for web layer: FormBean

Formbean is responsible for storing and transmitting the data of web layer and has the functions to verify the data input and reset the data control. The formBean must inherit from the ActionForm class. FormBean encapsulates the HTTP request as the object. For being easy to reuse and maintenance, many HTTP request can share a formBean.

The algorithms of code resource is following:

```
public class UserCheckActionForm
extends ActionForm {
  private String password;
  private String username;
  public String getPassword() {
return password; }//getter
  public void setPassword(String
password) { this.password =
password; }//setter
  ……
  public ActionErrors validate(……)
{……}//data validate
  public void reset(……) {}//reset
the control
}
```

3.4.2 Data expression for business layer: *VO*

VO is independent java object and responsible for storing and transmitting the data of business layer and has the methods of getter and setter. The format is similar with FormBean. The algorithms of code resource are following:

```
public class UserInfoBean {
  private String staff_id;
  ……
  public String getStaff_id() { return
staff_id; }//getter
  public void setStaff_id(String
staff_id) { this.staff_id =
staff_id; }//setter
  ……
}
```

3.4.3 Data expression for persistent layer: *PO*

PO class also is similar with VO, the code resource is omitted. The Key point is the mapping document, which includes the meta data required by ORM. The meta data is the PO statement and attribute mapping with the database.

4 Conclusion

This article has proposed a blue print to solve the problem of the data expression at the e-commerce website at enterprise level.

Based on the development architecture composed by the Struts and the Hibernate, it passes the data by the value object which the layers corresponding to, and strictly controls the visit to the persistence layer by the users. In this way, it can protect the business data effectively. Besides, it is useful to the upper debug and maintenance by separating the business logic and the data expression, and also by separating the data in different layers.

With the development of the Internet, it is absolutely necessarily for a large scale enterprise to be informational. Thus the application foreground of

J2EE is very amplitude. And the data expression question in its architecture system is becoming a hot topic. More and more open source or non-open source units or organizes are wild about the research on it. Each layer has its architecture package, such as the Struts and Spring Architecture of the business logic layer, the JDO、Hibernate ORM package of the persistence layer and so on.

The sealed pattern of the data expression in each layer influents the efficiency of the project development directly. The most popular way is to use both the Struts and Hibernate architecture. The Struts emphasize particularly on the control of the business logic, and the Hibernate is mainly used to seal the business data. Many projects have indicated that this way is effective.

In the data expression domain, there are still lots of details waiting people to explore and dig. It believes that with the lapse of time, there will be more outstanding methods to solve the data expression problem smoothly, and optimize the system architecture.

Acknowledgements

This research is supported by the open fund from Jiangsu Provincial Key Laboratory of Computer Information Processing Technology, Suzhou University. The authors are grateful for the anonymous reviewers who made constructive comments.

References:

[1] **Parra-Fuente Javier ,Fernandez-Alarcon Marta, Joyanes-Aguilar Luis,***Universal Web Application Server - One server for any Web Application technology***. WSEAS Transactions on Computers, v 5, n 9, September, 2006, p 2019-2024**

[2] Bowers, John, Cecil, J. **Internet based frameworks for virtual enterprises**. *WSEAS Transactions on Information Science and Applications*, v 3, n 8, August, 2006, p 1601-1607

[3] Michael Girdley, *J2EE Applications and BEA WebLogic Server*, Publishing House of Electronics Industry. 2002

[4]Wu Wei, Lu Jian-de, Research of Layer Pattern on Developing of J2EE Application, *Microcomputer Development*, Vol.15， No.1, 2005.1, pp.125-127.

[5] Chuck Cavaness, *Jakarta Struts*, O'REILLY Publishing, 2004, 151-184.

[6] Bambara J.J, *J2EE Unleashed*, Beijing：China Machine Press. 2002.

[7] Zhu Xiaoyi, The 3-tier B/S Enterprise Information System Based on J2EE, *Journal of Taiyuan University of Technology*, Vol.36， No.1, 2005.1, pp.56-59.

[8] Zhao Kaiqin, Yu Weihai, Web Frame Technology Based on J2EE, *Modern Electronics Technologies*, No.1, 2005, pp.31-33.

[9] Jim Keogh. *J2EE Complete Reference*, Publishing House of Electronics Industry. 2003.

[10] Xue Bing, Cao Zuo-liang, Application of design pattern and data persistence layer framework in Web system, *Journal of Tianjin Institute of Technology*, Vol.20, No.1, 2005, pp.10-13.

[11] LI Wei-dong,SHI Hua-ji,LI Xing-yi, Study of enterprise application integration based on web services on J2EE platform, *Computer Engineering and Design*, Vol.36, No.1, 2005.11, pp.53-155.

[12] Zhao Ming, Ma Yongliang, Zhang Xiaoshuan, Design and development of an automated information gathering and issue system based on a small and medium-sized website, *WSEAS Transactions on Computers*, Vol.6, No.12, 2007, pp.1174-1176.

[13] Perez-Sorrosal,Francisco; Vuckovic, Jaksa; Patino-Martinez, Marta; Highly available long running transactions and activities for J2EE applications, *Proceedings-International Conference on Distributed Computing Systems*, 2007.

[14] Lin, Bo; Zhou, Ming-Hui; Liu, Tian-Cheng; Huang, Gang; Web container integration framework in J2EE application servers, *Journal of Software*, Vol.17, No.5, 2006, pp.1195-1203.

[15] Pei,Songwen;Wu,Baifeng;Zhu,Kun;Yu,Qiang, Novel Software Automated Testing System Based on J2EE, *Tsinghua Science and Technology*, Vol.12, No.Supply, 2007, pp.51-56.

[16] Zeng Bing; Peng,Changgen; Yang,Hui; Zeng,Kai; Jiang,Xiaoli, Solution to J2EE technology-based enterprise asynchronous communication, *Computer Engineering*, Vol.32, No.5, 2006, pp.252-254.

[17] Yang,Tao;Zhou,Zhibo;Ling,Li, J2EE quick development framework based on struts and hibernate, *Computer Engineering*, Vol.32, No.5, 2006, pp.83-85.

[18] Lin,Lin;Yao,Yu;Zhong,Shi-Sheng, Research on the J2EE-based product database management system, *Journal of Harbin Institute of Technology*, Vol.14, No.1, 2007, pp.106-113.

[19] Tang,Wenzhong; Xu,Xiaoman, Performance monitoring and optimization design of database access layer about J2EE management information system, *Journal of Beijing University of*

Aeronautics and Astronautics, Vol.33, No.10, 2006, pp.106-109.