

Design and Implementation of Middle Layer for DetachedCriteria Query Based on JSF and Hibernate Framework

Jia Chen, Zheqiong Yan, Rui Gu

Management Science and Engineering Department, Dalian Maritime University, Dalian, China

Email: chenjia@dlnu.edu.cn, yanzheqiong@gmail.com, gurui@dlnu.edu.cn

Abstract—With the rapid development of Internet, B/S architecture has occupied the dominate place in information system engineering. Countless frameworks have emerged to boost the development of B/S systems. But how to take advantage of the convenience and avoid being tied to the chariot of these frameworks has become a dilemma for programmers. The paper analyzed the shortage of general query in layered structure, combined with idea of off-line query from Hibernate framework, named DetachedCriteria, proposed a new middle layer to accommodate the needs of off-line query to avoid the restrictions in the use of DetachedCriteria query under the Spring framework. The work principle and structure of the DetachedCriteria middle layer were illustrated using a real project on food origin tracing.

Index Terms—JSF, Hibernate, Off-Line Query, DetachedCriteria, Middle Layer

I. INTRODUCTION

With the rapid development of Internet technology, more and more information systems are based on the B/S architecture. In this application background, there is an increasing demand for J2EE and performance requirement is also rising. In J2EE, JSF has a great advantage for user interface, due to its component event model. JSF makes a clear distinction between business logic and presentation logic by providing a set of perfect mechanism to model events processing and save a component state in a certain time [1]. With the help of Hibernate framework, the integration framework can develop a flexible, low coupling, easy to modified information system rapidly. Thus the whole system can be built on a layered architecture with view layer, business layer, control layer and persistence layer [2].

Nowadays, the query requirement is not only the high query efficiency for user but also for programmer, how to develop an easy to maintain and expand query platform on the premise of ensuring the layered architecture. The traditional inquires make a strong dependence between web layer and data access layer, query values and query logical have to be transferred from web layer to data access layer, destroying the clear boundaries between two layers [3]. In order to solve this problem, the Detached

Criteria, put forward by Hibernate framework, appeared in programmer's eyes. The DetachedCriteria query makes the whole system layers clear, decreases the coupling while has high query efficiency. Unfortunately, the DetachedCriteria is a new feature, introduced by Hibernate 3. Expanding new feature in a mature framework will result in a lot of new problems when integrating with other framework. DetachedCriteria is an example; it is not compatible when using DetachedCriteria in Spring framework [4]. This circumstance will restrict the function of DetachedCriteria. This is also the significance of the existence of middle layer for DetachedCriteria [5]

II. THE NECESSARY OF MIDDLE LAYER INTRODUCTION

A. The Necessary of Integrating JSF and Hibernate

According to the knowledge of JSF and Hibernate, it's clear that JSF framework based on MVC design pattern isolate the sources program and business processes by controlling the configuration file, thus system is easy to be maintained and modified.

Though stand out in the view layer, JSF still leaves much to be desired in the model layer. Compared with JSF, Hibernate has many advantages in model layer, it's basically an ORM tool used for OR mapping which allows you to perform database activates without bothering about the Database change. But if only develop system using Hibernate, the logic of system is still confused, the process and view don't be separated effectively, as a result, the coupling degree isn't be decreased [6].

As a solution, integrating JSF and Hibernate is proposed. JSF, the infrastructure of whole system, is in charge of implementing the MVC pattern [7], and Hibernate is used to support the persistence layer. Thus, the coupling degree can be decreased by JSF and Hibernate is responsible for reducing the difficulty of processing business and accessing database, so that it's convenient to interactive with database. Based on this, the integrating framework can be useful and convenient in theory, but how to integrate two framework and transfer data between them in practice is becoming a main issue.

The Middle Layer is to solve this problem. As the initial aim of integrating is to simplify the process of accessing database while sticking to decrease the coupling degree and definitude multilayer framework, so the Middle Layer is a MVC-based design.

B. The Shortage of Traditional Query Mode in The Layered Architecture

In the traditional query mode, there are a large number of dynamic conditional query in the conventional Web programming, in other words, the user inputs some values into web forms, and then according to the user's selection criteria, the procedures dynamically generated SQL statements to search. As for the layered system applications, the query values need to be transferred from web layer to business object, and then the business object constructs select statements by listing out the values. It is traditional to use Map as a container of values [8].

However, the Map can only transfer the name and value, not deliver the logic of values that is conditional expressions, like greater, less than or equal. Thus, the business object has to involve many conditional expressions. So once the logic of values change, the business object query construction algorithm must be modified accordingly, but this query change is the implicit contract, rather than the code constraint, so it is prone to error. The process of using Map to pass a list of query values is shown in Fig. 1.

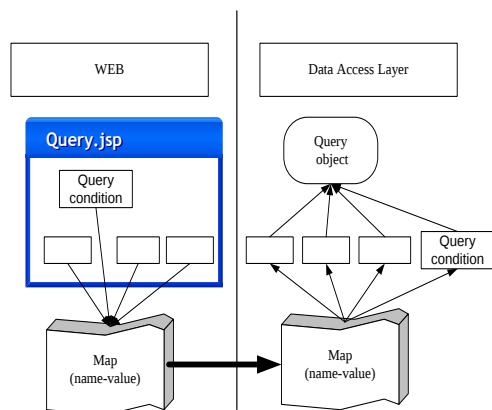


Figure 1. The process of using Map to pass a list of query values.

C. Concept of DetachedCriteria

To solve the above problems, this paper proposes the DetachedCriteria query mode, known as off-line query, introduced by Hibernate 3. The DetachedCriteria query mode and Criteria are very similar except the time to be created. The DetachedCriteria object is created without a hibernate session, out of the session limit, which means you can create DetachedCriteria object in the web layer, and the query values and logic can be encapsulated into the DetachedCriteria object, then what only to do is to pass the object to the data access layer. It is important to know that the implementation of DetachedCriteria objects is still need to be completed in one session. Unlike the Criteria query, it is only required a session when executing the query not creating, before that all the query values and logic have been encapsulated into the

DetachedCriteria objects. Thus, there is no need to list out the value and logic in data access layer. This brakes the traditional way in which it is inevitable that the data access layer has many query values and logic, strengthens the hierarchical concept [9].

The advantages of DetachedCriteria query: the task of constructing select statements is completely entrust to the web layer, while data access layer is only in charge of executing query action and focus on maintain the data, has nothing with query values any more, so that the layer is more solid, and its duty is only to access database [10]. Once the query values or logic needs to be modified, only need to do is change the codes in the web layer, all this has nothing with the data access layer. This follows the design principle; alter little to adapt more change, brings more convenience to debug and maintain system. The process of using DetachedCriteria to query is shown in Fig. 2:

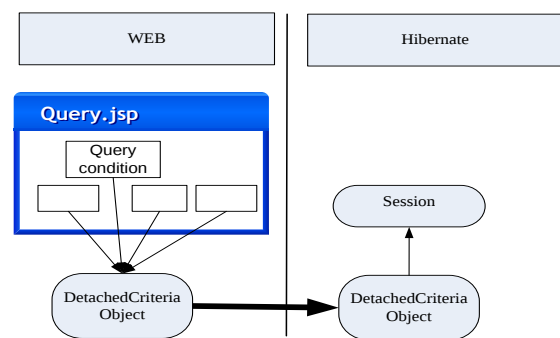


Figure 2. The process of using DetachedCriteria to query.

However, there is still a problem if only use the DetachedCriteria. According to the knowledge of DetachedCriteria, use the static function forClass(class) to instance a DetachedCriteria object. If create object in web layer, we have to add the statement, DetachedCriteria object = DetachedCriteria.forClass("tableName") in web layer.

This happens on the assumption that the web developer has already known the knowledge of DetachedCriteria and has a clear idea of which table should be access in the database. But the fact is not so. This is not only decoupling, on the contrary, increasing web layer and data access layer dependence, against the original intention of using DetachedCriteria [11].

D. The Restriction of Using DetachedCriteria in Spring Framework

To solve above problem, many middle-layer appeared. The more outstanding is Spring framework, which is a lightweight framework based on inversion of control (Ioc) and aspect-oriented programming (AOP). In the field of integrating framework, Spring has a good reputation, so now many systems have opted for the Spring. But as for the DetachedCriteria, the Spring leaves much to be desired.

Though Spring framework provides a function getHibernateTemplate (). FindByCriteria (Detached-

Criteria) to execute the DetachedCriteria query, according to development experience, the cast will throw an exception during the process. The reason is that HibernateTemplate's execute method provides a callback interface with the session as an argument. But in fact, the session which passed by HibernateTemplate to callback is a Proxy class belong the SessionImpl class, instead of the class org.hibernate.impl.SessionImpl. Unfortunately, the parameters of DetachedCriteria's method setExecutableCriteria are required to force the session into SessionImpl, while the Spring passed over a Proxy class, resulting in the restriction in handling DetachedCriteria in Spring framework. To play off the DetachedCriteria in Spring, the programmer has to know more about the source code of Spring, can handle it very skillfully, thus the process of developing system will be slow and difficulty is also rising.

Based on the above issues, the paper proposes a Middle Layer instead of Spring to handle DetachedCriteria query.

III. THE ISSUES MIDDLE LAYER TO SOLVE

According to the above statement, it is clear that the DetachedCriteria is the preferred choice in the hierarchical structure of the query system without the limitation of Spring framework. So the main issue Middle layer solved is to avoid the Spring framework, at the same time, ensure a clarity and low coupling hierarchical system.

In the JSF and Hibernate integration system, the Middle Layer plays a key role to handle DetachedCriteria, connects the JSF and Hibernate, while it also converts data type to balance the different between JSF and Hibernate's data type. Due to the Middle layer is designed for DetachedCriteria specifically; the function is simple relatively, so it can be used very easily, save a lot of time to learn Spring's source code. Of course, all this is built on the Middle Layer's high query efficiency and accuracy. The abstract process of using Middle Layer to query is shown in Fig. 3:

IV. THE DESIGN PRINCIPLES OF MIDDLE LAYER

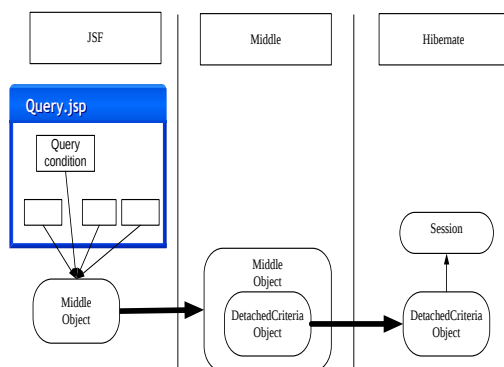


Figure 3. The abstract process of using Middle Layer to query.

A First principle is to provide interface rather than Implementation

In order to play DetachedCriteria's real role in hierarchical structure system, decouple the dependence between the web layer and data access, Middle Layer will encapsulate the DetachedCriteria object, web layer instance a middle object instead of generated a DetachedCriteria object directly. Thus, what the web developer needs to do is calling the interface of the middle layer, as for the middle layer is the DetachedCriteria object or something else, web developer does not know, neither need to know, just need to focus their energy on the web core functions. Similarly, for Hibernate data access layer, it only needs to accept DetachedCriteria object, and then complete the query action, as for how to collect the query values logic, it does not need to attend.

With the Middle Layer, the JSF and Hibernate don't know each other as much as possible; they carry out their own duties. View layer JSF simply do not know the Hibernate is responsible for the persistence in the system, and for Hibernate, it's not important whether the view layer is JSF or not.

In the integrated system, the extent of transparency between frameworks determines the ease of system modification. The more transparent, the more easy to modify and expand.

B. Open-closed principles: open for extension, closed for modification, the principle of design framework, that is, to minimize coupling.

The main function of Middle Layer is extract to core classes; the core classes implement transmission of the DetachedCriteria objects, and then call the data access class to execute the query action. Due to the query process is the same, which is to collect query values and logic, and then write the appropriate SQL statement to query the database. During the whole query process, often need to changed is query value and logic, which is DetachedCriteria object. Therefore, put the query process into the core class, which debugged many times to ensure the whole procedure is correct, then the core class should not be changed in the course of using. That is the closed principle.

According to the demand of query ,if need to add new query function ,only need to adding new query class on the basis of the original code. The new query classes are used as the old query classes do, but different data for the database table only. The whole framework after adding the middle layer is shown in Fig. 4.

In order to further understand the working principle of the Middle Layer, one example will be analyzed in detail.

V. THE IMPLANTATION OF MIDDLE LAYER

A Background

Food safety is a topic drawing prior attention of the society. As we all know, sub-standard food can cause the serious harm to the human body. With this in mind, Food Traceability System can track food information "from the

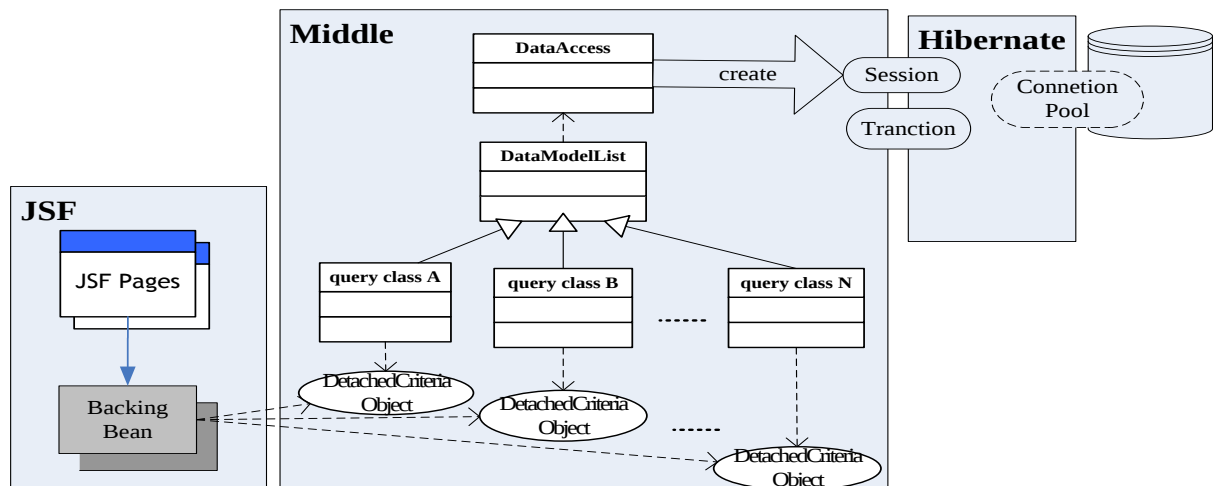


Figure 4. The whole framework after adding the Middle Layer.

origin to mouse". The core of Food Traceability System is tracing, identifying, and linking the information flow and real logistics to make all aspects of food traceable and identifiable. The data distribution of whole food traceability system is shown in Fig. 5.

In the food traceability system, due to regional differences in various aspects of the circulation, the distributed database should be considered. So, the top-level database and the underlying database are designed.

The underlying database set at various supermarkets and other retail outlets is responsible for managing the purchasing-selling-stocking data. The underlying database stored the food information throughout the traceability chain from the plant, production, and wholesale to retail, the traceability information to be passed through the RFID card among each link of chain. With RFID card, the basic information and food vendor information are passed from previous link to the next Link in traceability chain, thus ensuring the continuity of the whole chain, but also ensuring that every link of traceability information can be traced in the top database. The tables of underlying database contains: food coding table, purchase table, sale table, receive table and suppliers table.

While the top-level database, a query platform, is in

charge of summarize data from each underlying database for inquiring, aggregating and analyzing in the future. The data of top-level is collected from every underlying database by uploading data for a certain period time. Compared with the underlying database, the top-level database's data is no longer the sales information for the daily, but summary information within a period. It is very convenient to query and analyzed. The tables contains the information table each retail outlets, sales table.

At the same time, in order to make the query result is correct and the prediction is reliability, there should have a synchronization mechanism to maintain data consistency between the underlying databases and top-level database.

In this paper, the food traceability system terminals are installed in supermarkets, the customers use the system in the way of touch screen, and then system queries the top-level database through remote access to get traceable information from the plant, production, wholesale to retail throughout all aspects of retrospective basic information, so as to achieve the ultimate goal of traceability.

The requirement of query is when you feed a source tracing code in, the system will show the detail information and every aspects of this food's circulation

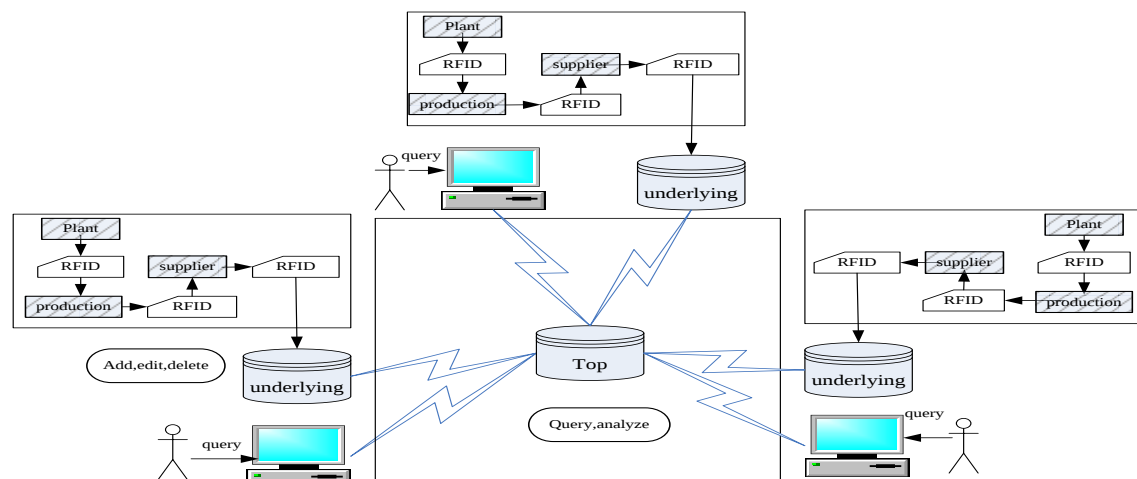


Figure 5. The data distribution of whole food traceability system.

information, and in other words, one operation involves two actions: query detail information and circulation information.

B Working Principle

First of all, the focus is the Middle Layer's working principles. In order to explain the working principle more clearly, we should analyze the class diagram. The class diagram of Middle Layer is shown in Fig. 6:

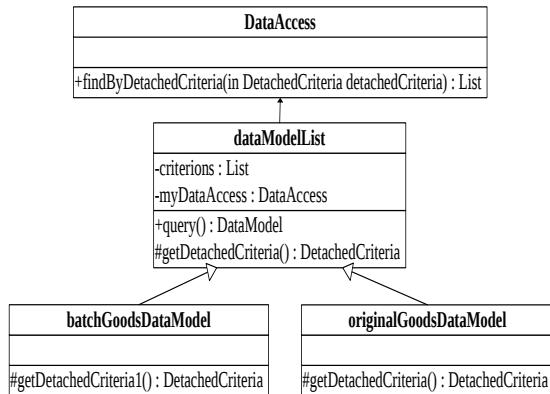


Figure 6. The class diagram of Middle Layer.

Middle Layer can be divided into three layers in concept: data access layer; core layer and query layer.

The data access layer includes DataAccess class, which is the interface with Hibernate. DataAccess is responsible for opening a Hibernate's session, as the life cycle of a query operation. It is also the bottom layer of Middle Layer, the other classes are based on this layer. The session will accept a DetachedCriteria object as a parameter, and then call the getExecutableCriteria (session), a method of DetachedCriteria class, to contract with the opened session. This process transfers an object from detached state to the persistent state, and then, in session, the object will be processed to query database data. The key codes are as follows:

```

public static List findByDetachedCriteria(DetachedCriteria object){
    List list=null;
    try{ session=HibernateUtil.currentSession();
        tx = session.beginTransaction();
        list= object.getExecutableCriteria(session).list();
        tx.commit();
    }catch (HibernateException e) {
        e.printStackTrace();
        tx.rollback();
    }finally{
        HibernateUtil.closeSession(session); }
    return list; }
  
```

The core layer is the second part of Middle Layer, including an abstract class DataModelList, which contains the main logic of the query action. In this class, instantiate a DataAccess object as a channel to access the database. Then using the polymorphism, return different sub-query object, inherited from DataModelList, as a parameter of

query() method., so as to achieve a different query requirement sharing a common query process.

Converting data types is another job for this class. In JSF, the components to display data are generally using the DataModel type as the data source, such as the DataTable. However, the data set returned from Hibernate is usually the List type. In order to display the query result in JSF pages conveniently and efficiently, we have to match these two types, convert the List type to DataModel type. That is what the DataModelList do in the function of transformation, achieving the easy use of the data set. This transformation function reflects the responsibilities of the Middle Layer, coordinating the different framework as much as possible, bringing the convenient to develop and maintain the system. The key codes are as follows:

```

public abstract class dataModelList {
    private DataAccess myDataAccess=new DataAccess();
    private DataModel dataList=new ListDataModel();
    private List<Criterion> criterions=
        new ArrayList<Criterion>();
    protected abstract DetachedCriteria getDetachedCriteria();
    final public DataModel query(){
        try {
            dataList.setWrappedData(myDataAccess.
                findByDetachedCriteria(getDetachedCriteria()));
        }catch (Exception e) {
            e.printStackTrace();
        }return dataList;}}
  
```

The third layer of the middle layer is the specific query class. Each class in this layer is a subclass inherited from abstract class DataModelList. Depending on the real query requirements, every query class encapsulates a DetachedCriteria object; it is also the interface between Middle Layer and JSF framework.

In these sub-classes, instantiate a DetachedCriteria object by calling the static method forClass (class). The argument of method comes from POJO class file generated by Hibernate, corresponding to the database table. When there is a new query requirement, add a new sub-class inherited from DataModelList, and then handle it in the same way handling other query classes, thus expand the query requirement.

All query classes are instantiated in the JSF's backing Bean according to different purposes; one instance of query class is a DetachedCriteria object. In the food traceability system, there are two query actions, so its need to design two query classes named batchGoodsDataModel and originalGoodsDataModel.

In the JSF framework, the Backing Bean collects the data from the JSF pages referenced on Backing Bean, So Bean can be seen as the logic part of the JSF.

As the requirement is relatively simple in the food traceability system, two query pages share a common Backing Bean named QueryBean. Meanwhile, the interaction with the Middle Layer is implemented in QueryBean. When a user clicked the query button, the

method `queryAction()` defined in `QueryBean` will be called. The major task of `queryAction()` is creating a `batchGoodsDataModel` object, actually, that object is a `DetachedCriteria` object encapsulating query values in the JSF pages. After collecting all query conditions, the method `query()` of `batchGoodsDataModel` object is called to return the results from `DataAccess` in the form of `DataModel`, thus, the component `DataTable` can use result as data source directly, without transforming. The key codes are as follows:

```
batchGoodsDataModel batchDataModel=
new batchGoodsDataModel();
if(this.batchNo==null || this.batchNo.isEmpty()){
    return "NullInput";
}
if(this.batchNo.length()<10){
    return "noRecords";
}
String batchLast=this.batchNo.substring(0, 10);
String goodsNo=this.batchNo.substring(10);
Criterion criterion1=
Restrictions.like("batchNo","%" +batchLast);
Criterion criterion2=
Restrictions.eq("goodsNo",goodsNo);
batchDataModel.getCriteria().add(criterion1);
batchDataModel.getCriteria().add(criterion2);
```

C The Advantage of Middle Layer

Using polymorphism, an import feature of the object-oriented programming, in abstract class `DataModelList` to return different off-line objects under the complicated application has many benefits. It can build a multi-layer Middle Layer with high reliability, maintainability, expansibility and reuse.

If the need of application is expanded, requiring to query more information, what we just need to do is to add a new query class inheriting from abstract class `DataModelList`, the new query class has to override the abstract method `getDetachedCriteria()`, then it can be used as the already existing query classes. The reason is the query logic is completely alike, except the query conditions. The key of Middle Layer is extracting the same code and logic to abstract class so that it can achieve maximum code reuse. A parallel case, if the query logic need to be modified, just change the implementation of the abstract class method, then test one query class, if one query class can be debug successfully, with the same way, other query classes can be adjusted thus, it really relents the programmers' modify task.

VI. SUMMARY

Nowadays, there is a prevalent trend of framework-based programming. Under this background, how to use framework flexibility and how to integrate different frameworks effectively have already become the major problems faced by programmers. Since every framework can not consider all future needs, so when using the framework, programmers should play the advantages of each framework fully, but not be limited by the existing framework. It is need to propose new idea when the existing framework can not meet the demands well.

In the background of inquires demand, the paper analyzed the shortage of general query in layered structure, combined with idea of off-line from Hibernate framework, named `DetachedCriteria`, proposed a new middle layer to accommodate the needs of off-line query to avoid the restrictions in the use of `DetachedCriteria` query under the Spring framework. The Middle Layer used in off-line query platform is based on JSF and Hibernate framework, applies the idea of multi-layer and polymorphism, so that it can be more reliability, more maintainability and more expansibility. In other off-line query application, the Middle Layer can be modified to adapt to complicated application.

The Middle Layer in practical applications is actually to bring a lot of advantages, such as enhance hierarchical thinking in integrating framework, increase the transparency between the two, the coupling of the whole system was reduced; provide the convenient to use `DetachedCriteria` query, simplify the query structure and query the logic of the work. But there are also some disadvantages, such as the core class is responsible for more functions, so the performance requirements and accuracy of the core class are higher; on the other hand, the demand for multi-table query, the Middle Layer needs further improvement.

ACKNOWLEDGMENT

Part of the research was sponsored by the Eleventh Five-Year-Plan Program for Educational Science, Liaoning Provincial Department of Education (Grant No. JG09DB088).

The compiling and publication of this paper was supported by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry of China.

REFERENCES

- [1] ZHOU Juan, LIU Gao-yuan, ZHANG Guo-ping (School of Information Engineering, East China Jiaotong University, Nanchang 330013 China); Developing Web Interface Using JSF[J]; Journal of East China Jiaotong University; 2007-02.
- [2] CAO Jing, GENG Xiang-yi (Dalian Jiaotong University, Dalian 116028, China); An Application Investigation Based on Integrated Architecture of JSF, Spring and Ibatis[J]; Computer Knowledge and Technology (Academic Exchange); 2007-13.
- [3] Design and Implementation of Remote-training Platform Management System[J]; Mecatronics; 2009-01.
- [4] LI Meng-fei, GAO Qi-juan, ZHAO Yu, SUN Li (School of Information and Computer, Anhui Agricultural University, Hefei 230036, China); Data analysis of detecting dissolved oxygen in water based on the frame of Grails+Hibernate+Spring[J]; Agriculture Network Information; 2009-04.
- [5] WANG Xiao-fei^{1,2}, JIANG Yu-ming¹ (1. College of Computer Science, Sichuan University, Chengdu 610064, China; 2. No. 78158 Unit of PLA, Chengdu 610051, China); Research on Framework of Military Logistics Information System Based on MVC[J]; Ordnance Industry Automation; 2010-02.

- [6] YANG Yang 1,2,TAO Liang1(1.School of Computer Science & Technology,Anhui University,Hefei,Anhui 230039,China;2.Dept.of Computer Science & Technology,West Anhui University,Lu'an,Anhui 237012,China);The Design and Implementation of the Network-aided Teaching Resources Management System Based on J2EE[J];Journal of Shijiazhuang University;2009-06.
- [7] XUE Ying,MIN Lian-ying,QIU Qiao-chun(College of Computer Science & Technology,Wuhan University of Technology,Wuhan 430063,China);Research on the Query Optimization Based on Hibernate Caching Mechanism[J];Computer Knowledge and Technology(Academic Exchange);2007-17.
- [8] HONG Ying-han , LIU Zhu-song , LONG Gui-he (Faculty of Computer, Guangdong University of Technology, Guangzhou 510006);Design and Implementation of Dynamic Form Based on SSH Framework[J];Modern Computer;2009-09.
- [9] Wang Qinmin Wu Xiuling(Library,North University of China,Taiyuan 030051,China);The Design on Virtual Reference Services System of Weapons Techonology Document Platform[J];Journal of Modern Information;2009-12.
- [10] WANG Ping,YOU Qiang-hua,WANG Xia (1.North Sichuan Medical College,Nanchong 63700;2.Chengdu Medical College,Chengdu 610083,China);Research and Implementation of CPR Interactive System Based on J2EE[J];Journal of Chengdu Medical College;2008-02.
- [11] ZHANG Chao,LI Chun-ping(School of Software,Tsinghua University,Beijing 100084,China);Improvement of search based on O/R MAPPING technology[J];Journal of Computer Applications;2006-02.

Jia Chen (Ph.D.) is a professor and master candidate supervisor major in management information system at Dalian Maritime University. She has published several textbooks on information management and information system by major publishers in China. She is also an experienced information system expert that actively participating in information-system project supervising and being in charge of system development for provincial projects.

Zheqiong Yan is a graduate student at Dalian Maritime University. Her major is Management Science and Engineering, and the main research interest is information system development.

Rui Gu is a lecturer at Dalian Maritime University. He graduated from Tokyo Institute of Technology with a Ph.D. in 2008. His research interest covers information system development and currently focuses on virtual community studies.