Research and Design of the Lightweight J2EE-based CRM System for an Enterprise

Lu Lv
Department of Computer
Huangshi Institute of Technology
Huangshi, Hubei Province, China, 435003
Binggan4321@sina.com

***Abstract*-This thesis studies the development of lightweight J2EE-based CRM(Customer Relationship Management) system. With reference to the function modules of the mainstream CRM system, through the analysis of customer needs, this thesis studies the implementation methods of CRM system of an enterprise on the J2EE platform, and completes the detailed design and programming of the CRM system. It focuses on the application project and values of the J2EE technologies (Three frames of Struts, Hibernate and Spring) in the CRM system.**

Keywords-Customer relationship management, J2EE, three-tier structure

I. RESEARCH BACKGROUND OF CRM

As a part of the Internet strategy of the enterprise, CRM(Customer Relationship Management), ERP(Enterprise Resource Planning)and SCM (Supply Chain Management), etc. together constitute the core competitiveness of the enterprise in the network era. The emergence of CRM is closely related to the evolution process of marketing approaches, with the integration of information technologies and decision support techniques, emphasizing internal and external interaction between the business and customers. Each step of revolutionary evolution is a result of combined effects of contemporary information technology development and new marketing strategies. CRM is the result of the evolution of marketing methods. The emergence of this new business model brings the transition from traditional 4P (Product、Price、Place、Promotion)to 4C(Customer 、 Cost 、 Convenience 、 Communication) in marketing modes.

II. THE THREE-TIER STRUCTURE OF J2EE TECHNOLOGY

J2EE is a system which makes use of Java2 platform to simplify the development, deployment and management of enterprise information system. The J2EE technology is based on the core Java platform, for developers, they only need to focus on the business logic and architectural design of various application systems, as for the cumbersome programming at the bottom, they can be collocated with different development platforms to make the development and deployment efficiency of application system significantly increase. The involved assemblies and API in the CRM system are Servlet, JSP, JDBC API and XML.

The system is designed in strict accordance with MVC mode, and is layered according to J2EE, so the interlayer is strictly divided into business logic layer, DAO layer and data persistence layer, etc., as shown in Figure 1. The persistence layer is absolutely prohibited from getting access to the controller in MVC tier, even from participating in the realization of business logic.

The presentation layer use the traditional JSP technology, but JSP script is prohibited in the pages in order to avoid making the JSP page get messy. JSP technology combines with the tag library of Struts, so that the presentation layer of the application is distinct with an excellent readability.

III. CRM SYSTEM'S TECHNICAL ARCHITECTURE

This system involves three frameworks: Struts, Spring and Hibernate, using JSP as the technology for presentation layer. The four techniques band organically together to build a robust J2EE application. Figure1 shows the framework of each layer and the cooperative work of JSP components.

IV. OVERALL PROJECT DESIGN

A. system function module

The system has five major functions of customer resource management, marketing management, service management, knowledge management and system management. The system has 19 modules of potential customers, clients, contacts, customer feedback, products and services, client review, my product, my quotations, sales opportunities, business activities, my contract, my documents, user management, role management, departmental management, system management, my schedule, my projects and personal information.

The division of the system background modules and the functions of each module are in Table 1. CRUD represents the database operation of creat data, retrieve data, update data and delete data. EXCEL derivation means to derive the list to the local computer in the form of Excel.

B. Process Flow

Take the process of customer information management module as an example, as shown in Figure 2:

C. Database Table Design

Take the role table for example, as shown in Table 2

Published by Atlantis Press, Paris, France.

D. The relationship between data structures and procedures

The architecture of this system is divided into four levels: the presentation layer, business logic layer, data persistence layer and database layer. Data is organized in the following ways at all levels:

1)The task of data sheet in the database layer;

2) The task of PO in the data persistence layer, RW data from the database through JDBC, and then in the CGLIB of Hibernate, transform the data from the database into the PO object through the O / R Mapping (Object-relational mapping) between the PO object and data sheet, according to the foreign key relationships among the sheets, transform the PO into POJO object for the utilization in the business logic layer;

3) POJO operates between the persistence layer and business logic layer;

4) DTO (data transfer object) operates in the business logic layer and the presentation layer, it is used to transfer data between the business logic layer and presentation layer;

5) FORM object operates in the presentation layer, responsible for the data transmission between the presentation layer and Servlet.

The relationship between them is shown in Figure 3:

V. SYSTEM IMPLEMENTATION

A. .Hibernate persistence layer

By using the Hibernate persistence layer, traditional JDBC operation database can be avoided and the database can be operated in an object-oriented way to ensure that the entire software development process is conducted in an object-oriented manner.

One task of achieving the system is to set up a data sheet according to the system requirements, so it will be easy to design the persistence object based on the data sheet. This system has a total of 25 data sheets, so we need to generate persistent classes and configure mapping files aiming at the 25 sheets. In the MyEclipse development tool, use the Hibernate Reverse Engineering tool to generate the Hbm.xml mapping file.

B. achieve the DAO layer

In the Hibernate persistence layer, we can use DAO assembly to re-pack database operations. Through the DAO layer, business logic lay and specific persistence layer will be separated. Once we need to replace the persistence layer, the assemblies in the business logic layer do not need any changes. DAO mode has three parts of DAO factory class, DAO interfaces and DAO interface implementation class.

The applied DAO assembly is based on Hibernate and Spring, generated and managed by the Spring container. Spring container is responsible for supplementing the DAO components with the basic SessionFactory which is needed for its operation. Spring supplies a lot of utility classes for Hibernate integration, through the LocalSessionFactoryBean class, SessionFactory of Hibernate can be incorporated into the IoC container.

Lightweight J2EE framework takes the Spring Ioc container as the core, a connecting link between the preceding and the following. Its upward management comes from the action from the performance level, its downward management covers the business logic layer components, as well as the management of DAO objects needed in the business logic layer. With the help pf the two utility classes provided by Spring: HibernateDaoSupport and HibernateTemplate, DAO assemblies can be easily implemented. Hiberate provides the first capsulation for the persistent visits, and Spring simplifies the assess to the persistence layer on the basis of Hibernate[1].

All DAO implementation class must provide a reference to SessionFactory. All DAO components need to be injected with the SessionFactory reference. This application configures the DAO components of each module in the Spring configuration file of the corresponding model. Take the configuration of the DAO object of client model as an example:

```
<bean id="ClientDAO"
        class="com.xuyuan.client.dao.impl.ClientDAOI
mpl"
        singleton="false" autowire="byName">
        <!-- Use ref local inject the SessionFactory
intoClientDAOImpl -->
        <property name="sessionFactory"
          ref="sessionFactory">
        </property>
    </bean>
```

The DAO component configuration can only use the implementation class, but not the interface. The DAO implementation class of the system provides the setSessionFactory method for dependency injection of SessionFactory reference. This configuration file is not configured with SessionFactory bean, the configuration of dataSource and sessionFactory is in another file.

C. Achieve the Service layer

The Service components in the client module (customer management module) will be taken as the example. In the Service interface of the client module, IClientService interface, there are a large number of business methods definitions, whose realization is dependent on the DAO components. Since each business method involves multiple DAO operations, the DAO operation is the operation of individual data record, and the access of business logic methods needs multiple DAO operations, so each business logic method may need a number of records visits. The business logic components face the DAO interface for programming, allowing separation of business logic components from the implementation of DAO components. Therefore, business logic components only care about the implementation of the business logic, without concern for the implementation of data access logic.

Published by Atlantis Press, Paris, France.

D. Achieve MVC layer

The MVC framework in this system uses Struts, because Struts is the most robust and most widely used MVC framework. Using the Struts framework can improve the controllability of the system and ensure the stability and availability of the system.

The only requirement for the integration of Struts and Spring is to make Struts transmit the intercepted client requests to the bean of the Spring container. The system uses the DelegatingRequestProcessor integration strategy, it can avoid creating too many DelegatingActionProxy instances, and it can transmit the requests to the bean in the Spring container ahead of time.

To make Struts responsible for the creation of Spring container at the start of the application, the configuration file struts-config-xxx.xml file of each model should add the following configurations:

```
<plug-in
    className="org.springframework.web.struts.ContextLoaderPlugIn">
        <!-- Assign the Spring configuration file of xxx module -->
        <set-property property="contextConfigLocation"
            value="/WEB-INF/spring/spring-xxx.xml" />
</plug-in>
```

When use the DelegatingRequestProcessor integration strategy, there is no need to determine the implementation class of action[2]. Because DelegatingRequestProcessor directly transmits the requests to the Spring container. For the integration of Struts and Spring, the Action of Struts needs configuration in two places.

For the integration of Struts and Spring,， Action needs to be configured in two places:

1)Configure the corresponding Action in the struts-config.xml file.

2)Configure the actual Action in the Spring container.

The following will show the configuration of the controller, taking the client module as an example.

1) The Action configuration in the Struts configuration file: The struts-config.xml file configuration is relatively simple, and there is not much difference compared with simply using Struts application.

When a user requests for a clientlist.do, ActionServlet will intercept the request, then transmits the request to the Action named clientlist in the struts-config-client.xml. Therefore, you must add the clientlist mapping in the struts-config-client.xml file[3]. The mapping code is as follows：

```
<!-- Display the list of customer information -->
    <action path="/clientlist"

type="org.springframework.web.struts.DelegatingActionProxy"
        name="ClientForm" validate="false">
            <forward name="success"
path="/listClient.jsp">
        </forward>
    </action>
```

2) The configuration of Action in Spring: DeIegatingRequestProcessor Action is responsible for forwarding the Action to the cognominal bean in the Spring container, so we need to configure cognominal bean in the Spring container, and this bean is the actual implementation class of Action. Therefore, we must add the following code fragments in the configuration file spring-client.xml of Spring in the client module:

```
<bean name="/UI/client/clientlist" singleton="false"
    autowire="byName"
    class="com.xuyuan.client.action.ClientListAction"
>
        <property name="clientService">
            <ref bean="clientService" />
        </property>
</bean>
```

VI. SUMMARIES

Benefit from the distinctive feature of Java, "programming in one place, operating everywhere ", JSP applications are not limited and the operating system platform of the server has good portability. JSP applications can be developed, deployed or extended in any environment. When the software needs to change, excellent decoupled architecture will be able to keep as many parts of the software as possible, so change the realization of the software as little as possible in order to meet the changes of customer demands. This architecture is the J2EE layered architecture. In excellent layered architecture, the control layer depends on the business logic layer, but not coupled with any specific business logic component. They only have loose coupling with the interface: similarly, the business logic layer depends on the DAO layer, not coupled with any specific DAO component, but programming to an interface. If we conduct software implementation in this way, even if there are some changes in software, other parts will not change as much as possible.

REFERENCES

[1] Rod Johnson, Juergen Hoeller, Alef Arendsen, Thomas Risberg, Colin Sampaleanu. Spring Framework training high-level programming Chiang Translation [M]. Beijing: China Machine Press ,2006:324-400

[2] Nathaniel T. Schutta, Ryan Asleson. Ajax and Java Advanced Programming Yang Guang Translation [M]. Beijing: People Post Publishing ,2007:99-199

[3] Justin Gehtland, Ben Galbraith, Dion Almaer. Ajax cultivation of the Road [M]. Beijing: Electronic Industry Press ,2006:61-77 ,189-192 ,237-255

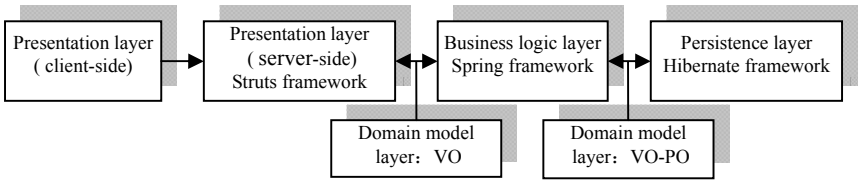

Figure 1. system layering and framework

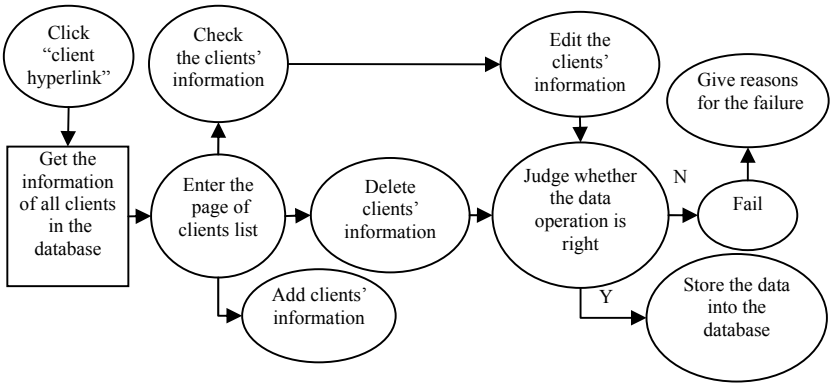

Figure 2. flow chart of client information management

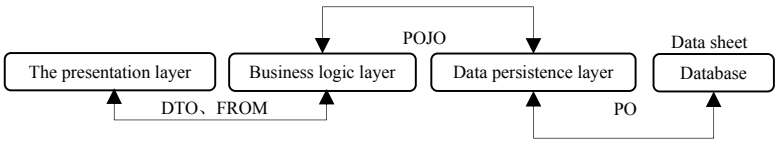

Figure 3. data structure and procedure relationship

TABLE.I SYSTEM BACKGROUND MODULE FUNCTIONS

Module	Categories	Features
User Management	System Management	CRUD/ EXCEL derivation
Role Management	System Management	CRUD/ EXCEL derivation
Sector Management	System Management	CRUD/ EXCEL derivation
Category Management	System Management	CRUD/ EXCEL derivation

TABLE.II ROLE TABLE DESIGN

Field Name	Data Type	Length	Keyword	Is Empty
RoleId	int	200	Y	N
RoleName	varchar	30	N	N
RoleDescription	varchar	500	N	Y