Developing Java EE Applications Based on Utilizing Design Patterns

Shuang Liu
College of Computer Science & Engineering
Dalian Nationalities University
Dalian, China
ls_liaoning@163.com

Peng Chen
Department of Computer Science & Technology
Neusoft Institute of Information
Dalian, China
chenpeng@neusoft.edu.cn

***Abstract*—Design patterns provide an efficient way to create more flexible, elegant and ultimately reusable object-oriented software. Thus their correct application in a system's design will significantly influence its reusability, flexibility, extensibility and maintainability. In this paper, we explain how design patterns are selected and implemented in a Java enterprise application naming Piloting management system based on a Model-View-Controller model and integration of open source frameworks such as Spring and Hibernate. Then we adopted several appropriate design patterns in the design process and validated the feasibility of this framework model in the implementation of Attemper subsystem.**

Keywords-design patterns; object-oriented; Java Enterprise Edition; UML; abstract factory

I. INTRODUCTION

More recently, as the internet has become more and more popular and play an important role in people's life, researchers have begun to focus on website development for the Internet Explorer web browsers. With the emergence of these sophisticated web site applications, maintaining flexibility and extensibility of the software has become extremely difficult.

Object-oriented software development [1] matured significantly during the past several years. The convergence of object-oriented modeling techniques and notations, the development of object-oriented frameworks and design patterns, and the evolution of object-oriented programming languages have been essential in the progression of this technology.

In the world of electric commerce with fast requirement changing, and the world of information technology, lower price, higher speed, littler resource using are required in enterprise program designing. Java Enterprise Edition (Java EE) [2] provided a designing method to develop, integrate and deploy programs based on Components, which intend to cut short development cost and trail enterprise application. The Java EE platform uses a multi-tiered distributed application model for enterprise applications, which make the components can be reusable in other applications. The Java EE platform also integrated with data exchange technique base on XML, which is a uniform security model and flexible transaction management.

The reason of utilizing design patters [3] are because patterns allow us to reuse solutions and establish common terminology. They give developers a higher-level perspective on the problem and on the process of a design and object orientation.

In this paper, we focus on developing an evolution framework model based on Spring and Hibernate complying with Java EE specification. In the process of design and implementing the framework, several appropriate design patterns are adopted and integrated with the practice feature of a Piloting Management Information System.

II. DESIGN PATTERNS

One of the most acclaimed benefits of object-oriented development is reuse, but it is one that requires management commitment to realize the benefits of reusing many artifacts of the development process. With design patterns as researchers' guide [4], they will learn how these important patterns fit into the software development process, and how they leverage them to solve their own design problems most efficiently.

Key features of patterns are their name, intent, problem, solution, participants and collaborators, consequences and implementation. When selecting a design pattern during the process of design and implementation, developers should consider how design patterns solve design problems, then scan their intent and study how patterns interrelate. After studying patterns of like purpose, they should examine a cause of redesign and consider what should be variable in their design.

Based on Unified Modeling Language (UML) mechanism, we show our choice of patterns in section 3.3 and implementation in section 4.

III. PILOT MANAGEMENT INFORMATION SYSTEM

Classes serve as the primary linguistic vehicle for reuse. Framework reuse and pattern reuse are at a higher level of abstraction than the reuse of individual classes and so provide greater leverage.

How to design the framework of the application, especially the implementation of its Java EE framework, and the reusability improvement of the application is the main topic of this section.

978-0-7695-3679-8/09 $25.00 © 2009 IEEE
DOI 10.1109/ICIE.2009.151

A. System Description

NingBo port is famous in the world with the daily average 60 piloted ships. All the jobs including the assignment of pilot task, driver task, and pilot boat task to the charge of piloted ships, the assess of pilots, vehicle drivers, pilot boats are finished manually at present. To improve the work efficiency of piloting management and saving human resource and material resource, NingBo pilot station decides to develop an enterprise application based on Java EE framework, which should be applicable for piloting management.

B. System Framework

The Java EE platform introduces a simplified programming model and eliminates much of the boilerplate that earlier releases required. Java EE implements the idea of Model-view-controller (MVC). MVC [5] is an architectural pattern used in software engineering. Successful use of the pattern isolates business logic from user interface considerations, resulting in an application where it is easier to modify either the visual appearance of the application or the underlying business rules without affecting the other.

The Spring Framework [6] is designed from the ground up to make it easier than ever to develop server-side applications with Java Enterprise Edition. Spring is an open source framework created to address the complexity of enterprise application development. One of the chief advantages of the Spring framework is its layered architecture, which allows developers to be selective about which of its components developers use while also providing a cohesive framework for Java EE application development.

Hibernate [7] is an open source object/relational mapping tool for Java. Hibernate helps developing persistent classes following common Java idiom - including association, inheritance, polymorphism, composition and the Java collections framework. Hibernate not only takes care of the mapping from Java classes to database tables, but also provides data query and retrieval facilities and can significantly reduce development time otherwise spent with manual data handling in SQL and JDBC.

As for the implement techniques of Java EE framework included in the Java EE specifications, one of the most important techniques is EJB. However, along with the rise of open source projects, we have more excellent tools to implement a Java EE framework, such as Struts, Spring and etc. Compared with EJB, these open source tools are more simple, easy and flexible. So, we selected some fashionable open source framework such as Spring and Hibernate as the appropriate framework during our enterprise application development.

Components of a modified Java EE model are shown in Figure 1 based on Spring and Hibernate and our new MVC model implementing Java EE framework is shown in Figure 2.

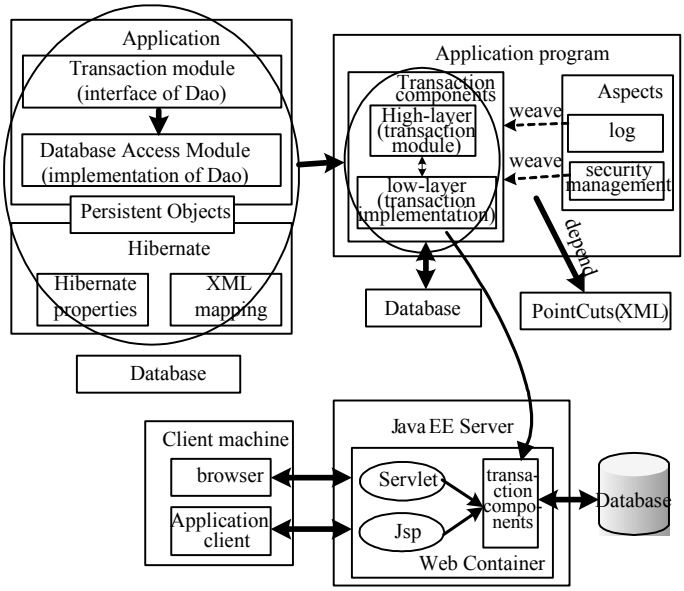

Figure 1. Implementation of business logic component by Spring+Hibernate.

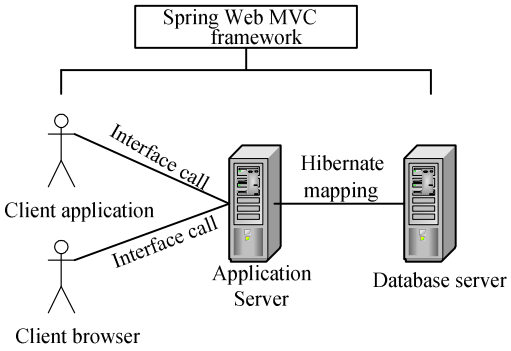

Figure 2. Java EE framework by Spring+Hibernate.

C. Design patterns selection

When selecting and utilizing design patterns, we follow the following steps:

Step1, study the pattern once through for an interview.

Step2, go back and study the structure, participants and collaborations.

Step3, look at the concrete sample code of the patterns.

Step4, choose names for pattern participants that are meaningful in the application context.

Step5, define the classes.

Step6, define application-specific names for operations in the pattern.

Step7, implement the operations to carry out the responsibilities and collaborations in the pattern.

In the implementation section, we use abstract factory as an example to explain how to apply design pattern in the development.

IV. IMPLEMENTATION

We use the module of basic data maintenance in the piloting management information system as an example to

explain how to instantiate the framework and how to apply design patterns in the software development process.

In the client side, resource lists are as following:

① Spring context resource file(ApplicationContext-client.xml), which provides transaction logic interface object to the client through dependency injection including url address, port number and name of the context of the server. At the same time, dynamic proxy tool org.springframework.aop.framework.ProxyFact-oryBean provided by Spring AOP is used to weave the privileges of every transaction service interface.

② Client window application, which gets transaction interface object through Spring context resource file and sends requirements to the server by calling the corresponding interface based on the user input.

In the server side, resource lists are as following:

① basic data maintenance: transaction interfaces (Manager) and its implementation (ManagerImpl) and their object/relational mapping file. We set log, security check here too. An example of ApplicationContext-service-basic.xml is shown as follows:

```
<bean id="shipOwnerManager" class="org.spring-framework.aop.framework.ProxyFactoryBean" >
<property name="target">
<bean class="com.nbpilot.service.basic.impl.ShipOwner-ManagerImpl">
  <property name="shipOwnerDao" ref="shipOwnerDao"/>
</bean>
</property>
……
</bean>
```

②basic data maintenance: data access interface (Dao) and its implementation (DaoImpl) and their object/relational mapping file names as applicationContext-dao-basic.xml. The DAO pattern abstracts and encapsulates all access to the data source. The implementation classes of it are DaoImpl that contains Hibernate-specific logic to manage and persist data. Using the xml file, dao interface corresponds to its implementation class by dependency injection.

③In order to deploy service provided by basic data maintenance transaction interfaces in the application server, file applicationContext-remote-basic.xml must be configured. This file sets definitions for every service name of every interface in the server. Then client sides need to initialize the calling interface object by the same service name when they access transaction interface object based on context resource file applicationContext-client.xml.

④Hibernate object/relational mapping file(*.hbm.xml), each table in the database needs a corresponding mapping file. For every table and its mapping file, the function is to define relationship between the property of the object and field of the table in the database. Spring Dao realizes conversion of objects and table records by these mapping files. We use ShipOwner.hbm.xml as an example,

```
<hibernate-mapping>
<class name="com.nbpilot.model.basic.ShipOwner" table="ship_owner" schema="NBPILOT" >
<cache usage="nonstrict-read-write" >
<id name="shipOwnerNum" type="integer">
  <column name="SHIP_OWNER_NUM" />
  <generator class="increment" />
</id>
<property name="shipOwnerCode"type="string">
 <column name="SHIP_OWNER_CODE" length="10" not-null="true" />
</property>
……
</class>
</hibernate-mapping>
```

⑤ Spring context resource file applicationContext-resource.xml, which configures dependency injection of the Hibernate data source including database type, driver program, url address, user name and password.

⑥web.xml file. In the Web MVC framework, users must connect to the front controller instead of required resource. Then front controller decides which requirements should be dispatched to which controller object. Here, org.springframework.web.servlet.Disp-atcherServlet plays the role of front controller. Users can implement their customized Servelet through file org.springframe-work.web.cont-ext.ContextLoaderListener as follows:

```
<listener>
  <listener-class>
      org.springframework.web.context.Context-
      LoaderListener
  </listener-class>
</listener>
```

According to the Gang of Four, the intent of the abstract factory is to provide an interface for creating families of related or dependent objects, without specifying their concrete classes. Class diagram of this pattern is shown in Figure 3.

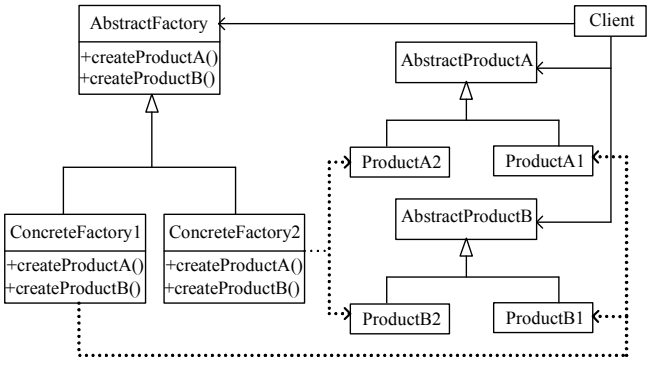

Figure 3. Class diagram of abstract factory.

The solution of abstract factory coordinates the creation of families of objects and gives a way to take the rules of how to perform the instantiation out of the client object that is using these created objects.

Participants includes: (1) AbstractFactory declares an interface for operations that create abstract product objects. (2) ConcreteFactory implements the operations to create concrete product objects. (3) AbstractProduct declares an interface for a type of product object. (4) ConcreteProduct defines a product object to be created by the corresponding

concrete factory and implements the AbstractProduct interface. (5) Client uses only interfaces declared by AbstractFactory and AbstractProduct classes.

Following steps of section 3.3, we use abstract factory pattern in the dao interface shown in Figure 4. It captures our strategic decisions regarding the structure of the classes.

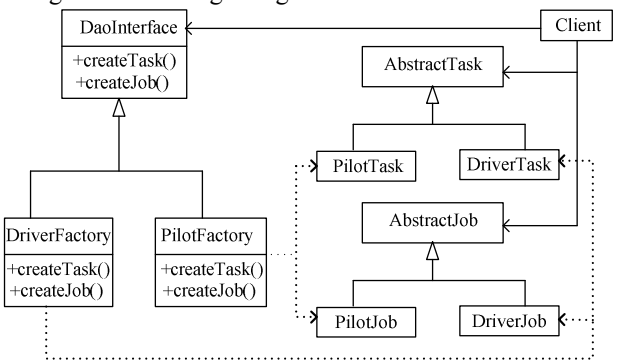

Figure 4. Class diagram of abstract factory application.

As we did for the DaoInterface interface and its subclasses, we design the most important elements of whole system in the architectural development process and evolve in detail over time in the coding process.

V. CONCLUSIONS

The enterprise platforms are complex as the Internet evolves. In this paper, we focus on how to develop object-oriented software with well-cohesion, coupling and completeness. With UML, design patterns revealing and reinforcing good object-oriented principles are utilizing to improve flexibility and reusability of the software. Because the Java EE platform introduces a simplified programming model and eliminates much of the boilerplate that earlier releases required, we propose an integration model based on Spring and Hibernate.

REFERENCES

[1] G. Booch, A. M. Rober and W. E. Michael, et al. Object-oriented analysis and design with applications, Peking: Posts & Telecom Press, 3rd ed, 2008.4, pp. 1-20.

[2] JavaTM 2 Platform Enterprise Edition Specification, Version: 1.4. Release: November 24, 2003, http://java.sun.com/.

[3] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design patterns-Elements of reusable object-oriented software, Peking: China Machine Press, 2004.9, pp. 1-29.

[4] A. Shalloway and J. Trott, Design Patterns Explained: A New Perspective on Object-Oriented Design, 1st ed., Peking: China Electric Power Press, 2003.7, pp. 80-86.

[5] W. Pengfei and H. Zhiming, “Application of Three Popular Open Source Software in MVC”, Journal of Jiangxi University of Science and Technology, 2006, 27(6), pp. 35-38.

[6] R.d Johnson, J. Hoeller, et al, Spring Reference Documentation (Version1.2.6), http://www.springframework.org/.

[7] X. Xi, Head First Hibernate, Peking: Publishing House of Electronics Industry, 2005, pp.1-16.