J2EE Development Based on the JSED Template

Hui Li[1], Hong Zhang [1], Yan Li[1], and Tiefeng Jin[2]
1 School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, Jiangsu Province, 221008 P.R.China,
lihuicumt@126.com, hongzh@cumt.edu.cn,
WWW home page: http://www.cumt.edu.cn
2 Information Management Division of Xuzhou Local Taxation Bureau,NO.120 South Zhongshan Road, Xuzhou, Jiangsu Province, 221000 P.R. China
jtf123@263.net,
WWW home page: http://www.xzds.gov.cn

Abstract. This article proposes the JSED template which is directed by the MVC design pattern and behaves as the Session Facade design pattern, and then gives its working principle, finally introduces how to use the JSED template in coding in actual J2EE developing environment.

1 Introduction

With the rapid development of Internet, J2EE component developing technology has become more and more widely applied in Web information system development by its mature features such as open, inter-platform and security. The MVC and the Session Facade and such J2EE basic design patterns have become the prevailing way increasingly in J2EE component development. How to integrate the classical design patterns and convert them into practical developing template has got more and more attention.

Please use the following format when citing this chapter:

Li, H., Zhang, H., Li, Y., Jin, T., 2006, in International Federation for Information Processing, Volume 205, Research and Practical Issues of Enterprise Information Systems, eds. Tjoa, A.M., Xu, L., Chaudhry, S., (Boston:Springer), pp.193-201.

2 J2EE Platform

2.1 J2EE Architecture

J2EE is a multi-layer, distributed and Component-Oriented enterprise level application mode proposed by SUN [1], which is composed by a series of services, APIs and protocols that offer functional supports to the development of Web-based multi-layer application. In J2EE application system, these components can be divided by function, and distribute on different computers and stay at corresponding layers.

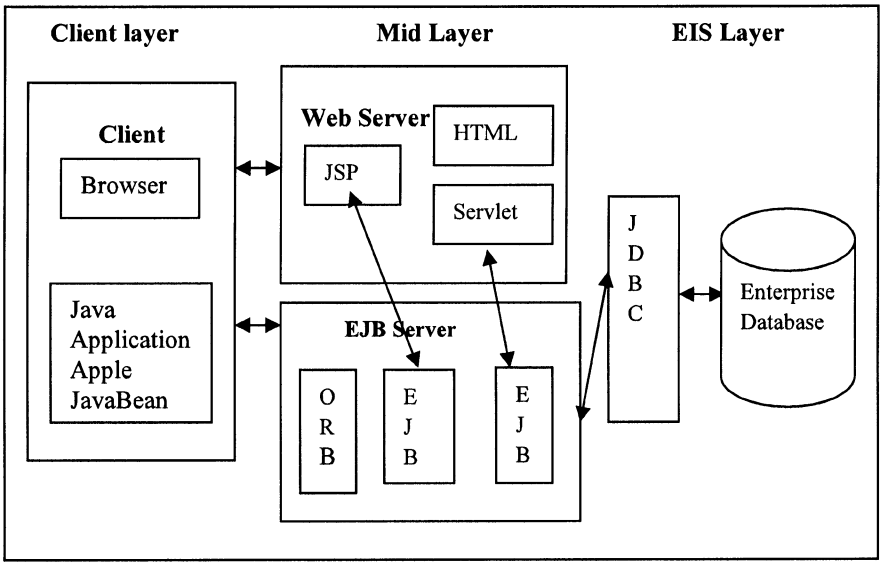

Fig. 1. Web Information System based on J2EE architecture

As shown in Fig. 1, J2EE is specially designed for creating multi-layer application. J2EE design is based on the following three layers: the Client Layer, the Mid Layer which also includes the Web Layer and the EJB Layer, and the EIS Layer. In the first two layers (the Client Layer and the Mid Layer), J2EE standard defines the components below [1]: Client components, Web components and EJB components.

2.2 EJB Technology

EJB is the core of J2EE. EJB (Enterprise Java Bean) is a category of object-oriented component, which is mainly used in developing, realizing and deploying distributed business logic, positioned at the Mid Layer of J2EE triad-layers. The core concept of EJB is to separate the business logic and the system logic of lower-layers so that the

developers just need to take care of the business logic and the system level service will be offered by application server (e.g. WebLogic Server) [2]. EJB 2.0 standard has defined three kinds of EJB as follows:

- Session Bean invoked by client components or Web components to accomplish server operations, e.g. accessing databases, invoking other EJB, etc.
- Entity Bean□on behalf of the persistent stored data, representative example is the records stored in databases.
- Message-Driven Bean□allowing EJB to receive JMS message asynchronously.

3 J2EE Design Patterns

3.1 The MVC Design Pattern

MVC is a comparatively successfully and widely used design pattern, particular suitable to design multi-layers application. MVC Design Pattern adopts a sort of idea of "Matrix Structure" which abstracts the application as Model, View and Controller [3].

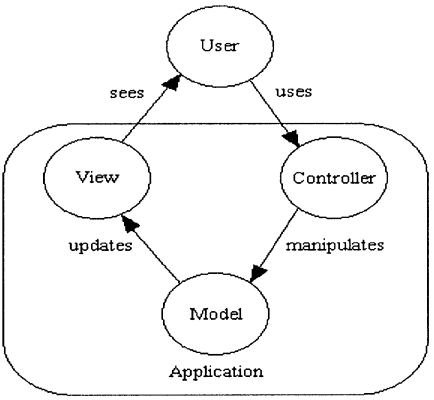

Fig. 2. The basic MVC relationship

Fig. 2 illustrates the relationship of the three parts in MVC Design Pattern [4, 5].As shown in Fig. 2, The Model, View and Controler are completely different in function in order to improve the quality of the software [6]. They are also intimately related and in constant contact. Therefore, they must reference each other.

In the Java language the MVC Design Pattern is described as having the following components [4]:

- An application model with its data representation and business logic.
- Views that provide data presentation and user input.
- A controller to dispatch requests and control flow.

The purpose of the MVC pattern is to separate the model from the view so that changes to the view can be implemented or even additional views created, without having to refashion the model [4].

3.2 The Session Facade Design Pattern

EJB design is the core model of J2EE component design, which emphasizes repeatability, maintainability, transplantable and so on. In the meanwhile, most of the J2EE applications contains data storage layer. Therefore, Client's accessing to Entity Beans is significant to EJB development. The simplest way to access Entity Beans is to access directly via client components, i.e. adding accessing code directly to the code of client components. But there are a great many hidden troubles in adopting this method, such as network load problems, exposing database substructure information, destroying transplantability of EJB, etc. In order to solve the above problems, in EJB design, it's necessary to avoid using client components and Web components to access Entity Beans directly. The best solution is to utilize the Session Facade Pattern in J2EE application system development. The Session Facade pattern realizes the Facade Design Pattern conception via using Session Beans, i.e. using Session Beans to pack Entity Beans to invoke methods of Entity Beans. An Entity Bean represents data in the database, and all operation to Entity Beans represents the change of substructure data. The synchronization process of Entity Beans and databases is managed by the container. Session Beans are used to process business logic and workflow, which is the abstraction of client work. Web layer is used to process the enterprise's present logic, i.e. processing the interaction between EJB layer and client layer, including reception and response of client requirements and sending the requirements to EJB and receive its response.

4 The JSED Template

4.1 The JSED Template Principle

EJB component is the core of J2EE. Although Java Servlet□Java Bean and JSP are not necessary components of J2EE application projects, they can work with EJB to offer application environment based on Java cohesion.

The MVC Design Pattern and the Session Facade Design Pattern are classical design patterns of J2EE application. In practical development, it should be guided by the basic pattern concept to design component development template that suitable to the actual situation. Component development has put the basic design pattern principle into practice, e.g. what components compose the three essential elements of the MVC pattern and the rule should be followed by the interfaces between these elements in actual development environment, etc. Proposal of the component template can greatly shorten component development cycle.

According to the analysis of the MVC and the Session Facade design pattern, the JSED (JSP-Session Bean-Entity Bean-Database) template is structured on the basis of integration and improving on these two design patterns. Following the main idea of the MVC pattern ,this template maps the three parts of the MVC pattern as different J2EE components, the Model part is mapped as the EJB components, the View part is mapped as the JSP-view components, and the Controller part is mapped as the JSP-controller components, i.e. inserting Java code into JSP to invoke EJB in the EJB server [7].While introducing the MVC pattern, the JSED template also uses the Session Facade pattern to organize and coordinate the invoking relationship between components in every part. Entity Beans access database components via JDBC; Stateless Session Beans access Entity Beans and executes business logic; JSP components (including JSP-view components and JSP-controller components)are used for showing data and submitting users' operation in order to accomplish view and control logic. In other words, the JSED template put forward by this article succeeds to the characteristics that the MVC and Session Facade patterns adapts to J2EE multi-layer application development.

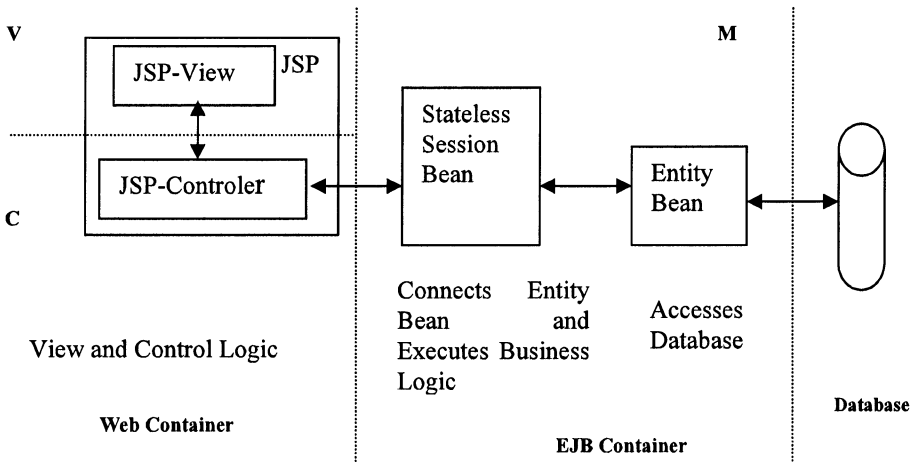

Fig. 3. Architecture of the JSED Template

A procedure here describes the specific steps about the JSED template.

(1) Various requirements from clients are delivered to JSP-Controller components via JSP-View components.

(2) According to the requirements, JSP-Controller components call EJB container for executing the corresponding business logic functions, for example, instantiating Session Bean objects, invoking Session Bean methods, updating or abstracting Session Bean objects' state.

(3) EJB container instantiates Session Bean objects as soon as it receives the requirements.

(4) According to the requirements from JSP-controller components, Session Beans execute the corresponding business logic functions, usually accessing Entity Beans via EJB container to operate the data. The results are looped back to the corresponding JSP-Controller components also via EJB container.

(5) JSP-Controller components deliver the results from Session Beans to the corresponding JSP-View components. These results are then formatted [8] by JSP-View components and looped back to clients by Web server.

4.2 JSED-based Coding Template

Directed by the JSED template principle, the specific coding template differs in the actual developing environment, which is mainly embodied in E-D link and S-E link. Here, following the D-E-S-J order, the JSED component coding template is given(the italicized parts need to be codified according to the actual circumstances).The actual development environment is described as follows: the database system is SQL Server 2000, the application server system is Weblogic Server 8.0, and the J2EE developing tool is JbuilderX.

(1) E-D Template (the template by which Entity Beans and Databases are connected)

- Create an EJB Model in the corresponding Project of JbuilderX to pack the interrelated Entity Beans and Session Beans, and then import a Database Schema to express the database which needs to be correlative with Entity Beans. In Database Schema Provider ,the database connection information should be set as follows:
  Driver: weblogic.jdbc.mssqlserver4.Driver
  URL: jdbc:weblogic:mssqlserver4:*Database Name@ DB Server Name*
  Username: *Login Name of SQL Server Identity Verification*
  Password: *Login Name of SQL Server Identity Verification*
  JNDI name: *Database JNDI Name Need to be Related*
  Other settings are empty
- Select a table in the imported Database Schema to create the corresponding Entity Beans (CMP is suggested). Entity Beans can be set and codified. It needs to be emphasized that the property named "Interface" should be set as "local", which means the local reference of Session Beans to Entity Beans.

(2) S-E Template (the template by which Session Beans and Entity Beans are connected)

- Create a stateless Session Bean in the EJB Model, and then create EJB Local References for the stateless Session Bean to access Entity Beans. The parameters are set as follows [9]:
  Name: *Local Reference Name*
  Select "isLink"
  Link: *name of the Entity Bean to be referred to*
  Type: Entity
  LocalHome: *Name of Database Associated With Entity Bean .Home Interface Name of Entity Bean*
  Local: *Name of Database Associated With Entity Bean. Entity Bean Name*
- In the source code of the Bean class, the coding template used to access Entity Bean is as follows:

```
public class Session Bean Name implements SessionBean {
  SessionContext sessionContext;
```

```
  private Entity Bean Name  Entity Bean instance Name =null;
  private Home Interface Name of Entity Bean  Home Object Name of Entity
Bean =null;
  public void ejbCreate() throws CreateException {
   try{
     Context context=new InitialContext();
     Object ref=context.lookup("java:/comp/env/ Entity Bean Name ");
     Home Object Name of  Entity Bean =( Home Interface Name of Entity
     Bean)ref;
    }catch(Exception e){e.printStackTrace();}
  }// access Entity Beans while creating Session Beans
  ..... //other call-back methods


  public void setSessionContext(SessionContext sessionContext) {
   this.sessionContext = sessionContext;
  }
 ..... //operation logic methods of Session Beans

 }
```

(3) J-S Template (the template by which JSP components and Session Beans are connected)

- Create a class named GetInitialContext ,and the coding template is as follows:

```
public class GetInitialContext {
 public  Context getInitialContext() throws Exception {
  String url = "t3://application sever name:7001";
  String user = null;
  String password = null;
  Properties properties = null;
  try {
   properties = new Properties();
   properties.put(Context.INITIAL_CONTEXT_FACTORY,
                  "weblogic.jndi.WLInitialContextFactory"); [10]
   properties.put(Context.PROVIDER_URL, url);
   if (user != null) {
    properties.put(Context.SECURITY_PRINCIPAL, user);
    properties.put(Context.SECURITY_CREDENTIALS, password == null ?
                   "" : password);
   }
   return new InitialContext(properties);
  }
  catch(Exception e) {
   System.out.println("Unable to connect to WebLogic server at " + url);
   System.out.println("Please make sure that the server is running.");
   throw e;
  }
 }
```

```
}
```

- Create JSP components to invoke Session Beans, and the coding template using for connecting Session Beans is as follows(the boldfaced"**GetInitialContext**" in the code is the class defined in the previous step),

```
<%@ page import="Project Name. Session Bean Name, Project Name. Home
  Interface Name of Session Bean, Project Name.GetInitialContext"%>
<% //invoke Session Bean facade
  Session Bean name Remote Interface Object Name of Session Bean =null;
  Home Interface Name of Session Bean Home Object Name of Session Bean;
   try{
    GetInitialContext ic=new GetInitialContext();
    Context ctx = ic.getInitialContext();
    Object objref=ctx.lookup("Session Bean name");
    Home Object Name of Session Bean =(Home Interface Name of Session
    Bean)PortableRemoteObject.narrow(objref, Home Interface Name of Session
    Bean.class);
    Remote Interface Object Name of Session Bean = Home Object Name of
  Session Bean.create();
    ...../
```

- The operation logic methods of Session Bean are invoked by the under mentioned template in JSP-Control components,

```
Remote Interface Object Name of Session Bean. Operation logic method Name
of Session Bean (Parameter Table);
```

5 Conclusions

In allusion to the practical application of the MVC and the Session Facade and other J2EE basic design patterns, we put forward the JSED template which is used in actual J2EE development.

In this paper, the principle and workflow of the JSED template are discussed, and the main code of this template is presented via the integration of the corresponding supporting tools and development environment.

It's obvious that it is more convenient to develop a web information system with legible structure, high performance, high security and high expansibility by applying the JSED template in J2EE development.

References

1. X. Yi, *An Instance Course for J2EE* (Beijing Hope Electronic Press, Beijing, 2002).
2. M. Girdley, R. Woollen, and S. L. Emerson, *J2EE Applications and BEA WebLogic Server* (Publishing House of Electronics Industry, Beijing, 2002).
3. R. Lu, Z. Yu, Y. Ruan, and Z. Wang, Study and Implementation of MVC Design Pattern on J2EE Platform, *Application Research of Computers* **20**(3), 144-146 (2003).

4. The Model-View-Controller (MVC) Design Pattern for PHP, Tony Marston (May 21, 2004); http://www.tonymarston.net/php-mysql/model-view-controller.html.
5. X. Zhang and Z. Teng, The J2EE Application Model JMVC Based on the MVC Design Pattern, *Application Research of Computers* **20**(9), 63-64 (2003).
6. M. Yuan, Y. Huang, J. Huang, and Y. Weng, The Research and Application of MVC Software Architecture Based on J2EE, *Application Research of Computers* **20**(3), 147-149 (2003).
7. H. Liu, Y. Li, and P. Su, The Research on the Distributed Web Application Based on J2EE Construction, *Application Research of Computers* **20**(9), 47-49 (2003).
8. C. Yang and B. Meng, The Construction of J2EE-based B2B E-commerce Platform, *Application Research of Computers* **20**(3), 140-143 (2003).
9. Q. Zhao and X. Qiao, *J2EE Application Development* (Publishing House of Electronics Industry, Beijing, 2003).
10. H. Zhang, *The Best book For Java 2 Enterprise Edition Programming By Example* (China Railway Publishing House, Beijing, 2002).