Cartoon Resource Management System Development Based on the Lightweight Java EE Architecture

Zeng Yongmin, Zhang Pingjian, He liangliang , Wan Kai, Zheng Qinqin

South China University of Technology, Guangzhou, 510006, China
pjzhang@scut.edu.cn

***Abstract*—The Java EE (Java Enterprise Edition, used to be called J2EE) platform is a multi-layer, distributed application framework that provides system level services to facilitate application development. This paper discusses the best practice of enterprise computing using the lightweight Java EE architecture and its applications in the development of a Cartoon Resource Management System. The system manages not only cartoon resources such as models created with Tools such as PhotoShop, 3D Max, and Autodesk, but also maintains related views of the model, thus, provides the user with visual aids when managing the resources.**

Keywords-Lightweight Java EE Architecture; Cartoon Resources; Management System; Dublin Core

I. INTRODUCTION

Enterprise computing is the distributed computing as practiced in medium-sized or large organizations that feature huge data, complicated business logic, dense transaction and high security ([1]).

The creation of Java EE platform is targeted exactly for enterprise computing and becomes prevalent in the e-government, e-commerce and MIS development community due to its advanced architecture, rich system level services, easy application development and deployment, and wide support from various vendors, organizations and communities ([2]). Nevertheless, prior to the specification of EJB 3.0, Java EE is notorious for its complicated design and high overhead, and people turn to the lightweight Java EE architecture consisting of simplified programming model and more responsive frameworks where open source projects play essential roles ([3]). Although with the advent of Java EE 5, these have been remedied extensively, lightweight Java EE architecture is still widely adopted especially for medium to small projects ([4,5]).

This paper analyzes some drawbacks of traditional Java EE architecture and pointes out advantages of adopting lightweight Java EE architecture. Finally, the design and development of a Cartoon Resource Management System is discussed in detail for demonstration.

II. LIGHTWEIGHT JAVA EE ARCHITECTURE

Traditional Java EE platform provides lots of system level services such as Bean life-cycle management, resource pooling, remote connectivity, multi-threading, JNDI service, persistency, and declarative transaction and security services which facilitates N-tier application development greatly. The Java EE platform has become the dominant player in the enterprise computing market since its advent. Fig. 1 shows the traditional Java EE model. 1 shows the traditional Java EE model.

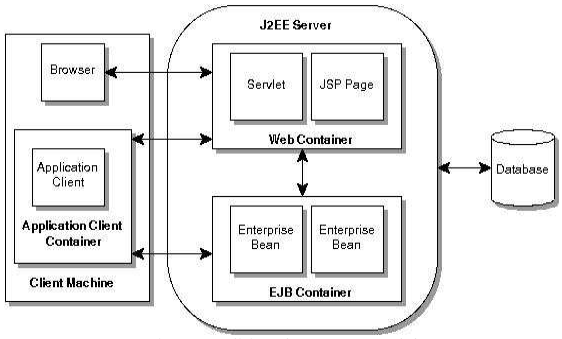

Fig. 1 Traditional Java EE model

Although Java EE architecture is an excellent solution for enterprise computing, some weaknesses have been identified after years of programming practices. Over the years, many excellent open source projects targeting at streamlining and simplifying enterprise computing have stemmed from Java communities, Struts, Spring and Hibernate are among the most famous and popular lightweight Java EE frameworks. Below are some brief reviews.

1) Struts ([6])

Struts is a famous MVC based web application development framework. Struts integrates Servlet, JSP, Tag library and application resources seamlessly and supports the MVC model via Struts Action, Struts Form Beans, Struts tag libraries, and Struts Controller to smooth the application logic.

2) Spring ([7])

Spring is an excellent POJO container that embodies many brilliant ideas including IoC (inverse of control) and AOP (aspect oriented programming). Since its release, Spring has become popular as an alternative, replacement, or even addition to the EJB model.

3) Hibernate ([8])

Hibernate is the most successful O/R mapping middleware that handles mappings between domain objects and database relations, bridging the gap between Java EE applications and DBMS. Hibernate solves the Object-Relational impedance mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions.

4) XDoclet

XDoclet is officially termed as "Javadoc metadata templating engine", it parses metadata in Java source files

978-0-7695-4608-7/12 $26.00 © 2012 IEEE
DOI 10.1109/ISdea.2012.597

and generates artifacts such as XML descriptors and/or source code. XDoclet is a natural choice when using Hibernate, as mapping files and database creation scripts can be automatically generated from tags in the Java object to be persisted.

5) JUnit ([9])

JUnit is a regression testing framework. Using JUnit as the unit testing suit for system development can simplify greatly the system testing procedure and workload. JUnit is an important tool in test-driven development and extreme programming.

The main advantages in adopting lightweight Java EE frameworks lies in

1) The combination of simple framework is power enough to compete the heavyweight EJB container.

2) The lightweight Java EE frameworks are easier for deployment of clustered services.

3) The lightweight Java EE frameworks reduce the complexity of the system and are more manageable and configurable.

4) With AOP support, the lightweight Java EE frameworks can provide more powerful declarative facilities and services.

It is worth pointing out that these lightweight Java EE frameworks have influenced strongly the evolution of Java EE itself. In fact, the EJB 3.0 specification and the J2EE 5 specification, for example, have already incorporated many features introduced in Spring and Hibernate.

III. Design of the Cartoon Resource Management System

The cartoon resource management system consists of the following components: the metadata management component; the cartoon resource management component and the user management component, each described as below.

1) The metadata management component adopts the Dublin Core metadata for cataloging cartoon resources. The Dublin Core set of metadata elements provides a small and fundamental group of text elements through which most resources can be described and catalogued. Using only 15 base text fields, a Dublin Core metadata record can describe physical resources such as books, video, sound, image, or text files, and composite media like web pages.

2) The cartoon resource management component maintains a collection of cartoon models created from tools such as PhotoShop, 3D Max, and Autodesk, together with corresponding effect images in usual graphic formats so that users can visually view the models. Equipped with the Dublin Core metadata, models can be searched via any combinations of the 15 metadata elements.

3) The user management component. This is a usual part that defines various roles, permissions, associates roles with corresponding permissions, and assigns appropriate roles to people, thus guarantees the secure access control.

In developing enterprise applications, design patterns play essential role for the system to achieve desired functionalities and qualities.

Design patterns are general reusable solutions to commonly occurring problems in software design ([10]). There already exist some proven design patterns such as the MVC pattern, the session façade pattern for the Java EE architecture.

In the requirement analysis and system design for the cartoon resource management system, 3 tiers are decoupled: the representation tier, the business logic tier, and the data tier, with large amount of mature, frequently utilized design patterns to simplify the complexity of design and development in each of the 3 tiers.

The architecture of the cartoon resource management system is designed as shown in Fig. 2 below.

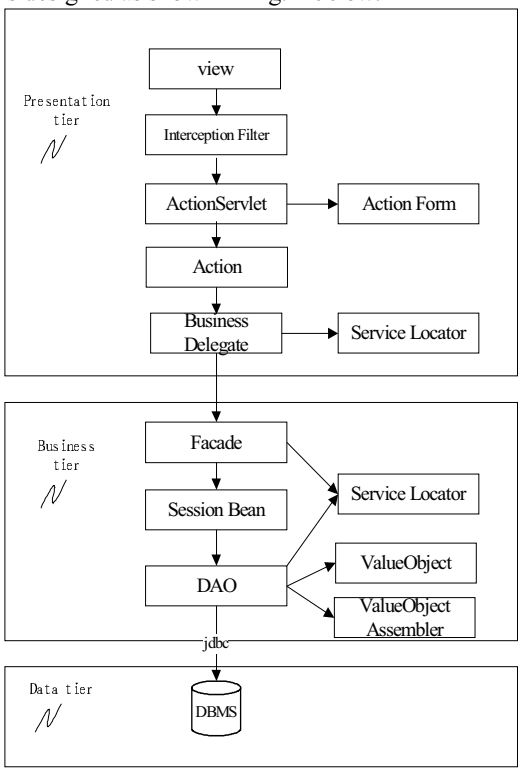

Fig. 2 Architecture of cartoon resource management system

IV. Implementation of the CCartoon Resource Management System

In this section, the implementation of the cartoon resource management system using lightweight Java EE frameworks is discussed.

The presentation tier is relatively easy and does not involve core business logics. Struts is used in the Web tier to streamline web development.

The business services are wrapped in the module Service, while data access services are wrapped in the module DAO. These two modules use POJO only and separate implementations from interfaces. Furthermore, the implementation classes of module DAO inherits from the HibernateDaoSupport, a Hibernate DAO template class provided by Spring.

Among lightweight Java EE frameworks, Spring takes over the responsibility of the "container" and "glue" together services with built-in support for Struts and Hibernate. For example, Spring integrates Hibernate readily by managing the Hibernate SessionFactory object, as well as Struts by providing ActionSupport.

A. Use of Spring IoC

Dependency among Java classes is described in the Spring configuration file applicationContext.xml as shown in the following snippet:

```
…
<bean name="/userInfo"
   class="com.sunrui.struts.LoadUserInfoAction">
        <property name="userInfoDao">
          <ref bean="userInfoDao"/>
        </property>
   </bean>
   ...
```

Fig. 3 Snippet of applicationContext.xml

```
public class LoadUserInfoAction extends ActionSupport
{
   private UserInfoDao userInfoDao;
   public void setUserInfoDao(UserInfoDao userInfoDao)
   {
      this. userInfDao = userInfDao;
   }
   //...
}
```

Fig. 4 LoadUserInfoAction.java

In the same configuration file there are also defined Hibernate transaction managers, Hibernate session factory bean and data source for JDBC connections.

B. Use of Hibernate

This is realized with strict relation to Spring. There are DAO interfaces defined with their implementations so that they can be used with the IoC pattern. As an example, the UserInfo entity has its interface and implementation pair on Fig. 5 and 6 respectively.

```
public interface UserInfoDao
{
   //...
   public UserInfo loadById(Long id);
   public void save(UserInfo instance);
   public void update(UserInfo instance);
   public void delete(UserInfo instance);
   //...
}
```

Fig. 5 UserInfoDao.java

```
public class UserInfoDaoImpl extends
   HibernateDaoSupport implements UserInfoDao
{
   //...
   public UserInfo loadById(Long id)
   {
         return (UserInfo)
      getHibernateTemplate().load(UserInfo.class, id);
   }
   public void save(UserInfo instance)
   {
      getHibernateTemplate().save(instance);
   }
   //...
}
```

Fig. 6 InfoDaoImpl.java

C. Use of Struts

Struts handles almost all business logic of the cartoon resource management system. Adding user data action shows how Struts works with Spring IoC (Fig. 7 and 8).

```
      /**
   * @struts.action input="/user/addUserInfo.jsp"
   validate="true" path="/user/addUserInfo"
   * name="UserInfoForm"
   * @struts.action-forward
   path="/user/listUserInfos.do" name="success"
   * redirect="true"
   * @spring.bean name="/user/addUserInfo"
   */
   public class AddUserInfoAction extends ActionSupport
   {
      private UserInfoDao userInfoDao;
      /**
       * @spring.property ref="UserInfoDao"
       */
```

```
    public void setUserInfoDao(UserInfoDao userInfoDao) {
        this.userInfoDao = userInfoDao;
            }
    public ActionForward execute(ActionMapping
        mapping, ActionForm form, HttpServletRequest
        request,HttpServletResponse response) throws
        Exception
            {
        UserInfoForm userInfoForm = (UserInfoForm) form;
        String username = (String)
            request.getSession().getAttribute("ACEGI_SECURITY_
            LAST_USERNAME");
        UserInfo userInfo = new UserInfo();
        userInfo.setHeader(userInfoForm.getHeader());
        userInfo.setUsername(username);
        if (isCancelled(request))
            return mapping.findForward("cancel");
         else {
            userInfoDao.save(userInfo);
            return mapping.findForward("success");
        }
    }
}
```

Fig. 7 AddUserInfoAction.java

```
<bean name="/user/addUserInfo"
    class="com.sunrui.struts.AddUserInfoAction">
    <property name="UserInfoDao">
        <ref bean="UserInfoDao"/>
    </property>
</bean>
<bean id="UserInfoDao" parent="txProxyTemplate">
    <property name="target">
        <ref bean="UserInfoDaoTarget"/>
    </property>
</bean>
<bean id="UserInfoDaoTarget"
    class="com.sunrui.dao.UserInfoDaoImpl">
    <property name="sessionFactory">
        <ref bean="sessionFactory"/>
    </property>
</bean>
<bean id="txProxyTemplate" lazy-init="true"
    class="org.springframework.transaction.interceptor.Trans
    actionProxyFactoryBean" abstract="true">
    //....
   <property name="transactionAttributes">
     <props>
        <prop
    key="loadOrCreate">PROPAGATION_REQUIRED
     </prop>
        //...
     </props>
   </property>
</bean>
```

Fig. 8 Snippet of applicationContext.xml

V. CONCLUSIONS

Developing applications on lightweight container is a promising solution compared to heavyweight ones with complex specification of EJB for small to medium applications, even for large scaled ones. The features which EJB specification offers can also be achieved by integrating specific frameworks. This is demonstrated by the development of a cartoon resource management system.

ACKNOWLEDGEMENT

This work is supported by the Guangdong Provincial Key Scientific and Technological Project under grant # 2010B010600020.

REFERENCES

[1] M. Fowler, Patterns of Enterprise Application, Addison-Wesley, 2002.

[2] R. Johnson, et al, Professional Java Server Programming: J2EE 1.3 Edition, Wrox, 2001.

[3] R. Johnson, J. Hoeller, Expert One-on-One J2EE Development without EJB, Wiley, 2004.

[4] T. Bak, B. Sakowicz, and A. Napieralski, “Development of advanced J2EE solutions based on lightweight containers on the example of "e-department" application”, *Proceedings of the International Conference on Mixed Design of Integrated Circuits and Systems, MIXDES 2006*, pp. 779-782.

[5] L. Duboc, T. Wicks, and W. Emmerich, “Experience with lightweight distributed component technologies in business intelligence systems”, *Software Engineering and Middleware - 4th International Workshop, SEM 2004*, 2005, p 214-229.

[6] T. Husted, *Struts in Action: Building Web Applications with the Leading Java Framework*, NetLibrary, Inc., 2003.

[7] C. Walls, and R. Breidenbach, *Spring in Action*, Manning Publications Company, 2007.

[8] C. Bauer, and G. King, *Hibernate in Action*, Manning Publications Company, 2005.

[9] C. Martin, *Extreme Programming in Practice*, Addison-Wesley, 2001.

[10] E. Gamma, et al, *Design Patterns: Elements of Reusable Object-oriented Software*, Addison-Wesley, 2004.