Research on e-Learning System Based on Java EE Architecture

Simeng Chen[a] and Bo Song[b] +

Software College, Shenyang Normal University, Shenyang 110034, China

[a]chensimeng63@126.com, [b]songbo63@126.com

Keywords: Java EE; JPA; ORM; JSF; EAO patterns and e-Learning.

Abstract. This paper proposes a solution of e-Learning system based on Java EE architecture. It combines EJB 3, JSF and open-source Ajax framework ICEfaces for Java EE applications to stratify reasonably. The rich UI components provided by ICEfaces are embedded in the presentation layer and the generic EAO patterns is introduced into the data persistence layer as data abstraction interface of business logic layer. It not only greatly simplifies human-computer interaction experience and running stability of NTS, but also has higher scalability and portability than using third-party ORM framework of e-Learning system.

Introduction

The lightweight J2EE application architecture based on open source framework of Spring, Hibernate and Struts, as with the advantage of development framework of loosely coupled, the operating system and database independence, low cost, easy to maintain and upgrade, has been widely applied to the field of network education [1]. But there are three problems of e-Learning system in the framework. First, although third-party ORM framework can use abstract data access code of DAO patterns to implement loose coupling with data persistence layer [1] [2], they are provided by different manufacturers. Also there is no unified standard interface. So the scalability and portability of e-Learning system are reduced. Second, in the data persistence layer, because type-safe interface cannot be provided in data access code, it will lead to a lot of repetitive compulsory type conversion work. Not only does it make bloated code, but also easily lead to runtime exceptions, which will affect the performance of system. Third, with the introduction of client programming of Ajax and Rich, the network teaching system which based on B/S structure can gradually provide more extensive end-user experience. Although this is user-friendly improvement, but the development of client programming of Ajax/Rich is related to the complex applications of the JavaScript programming [3].

For above problems, this paper proposes an e-Learning system solution based on Java EE architecture. The new scheme builds a Web page with EJB 3 JPA (Java Persistence API), and the JPA combines JSF and rich UI components of the ICEfaces [4]. Thereinto, ICEfaces provides Ajax-based components for Web interface in the presentation layer, EJB 3 JPA supplies support for data persistence, EJB Session Bean offers business logic services. This paper uses e-Learning system as a development case. It discusses two problems as following. Firstly is discussing how to design and implement the scalability, portability and flexibility of data persistence model based on EJB 3 JPA AND JSF technology. Secondly is discussing how to using pure-Java (not JavaScript) which is provided by ICEfaces framework to implement client-side Ajax functionality by RIA (Rich Internet Application).

JPA and EAO patterns

There are two reasons to import new JPA ORM specification [5]. Firstly is simplifying the development object persistence of existing Java EE and Java SE applications. Secondly is integrating

+ Corresponding author. Tel.: +861-394-0536713; fax: +86-024-86592390.
E-mail address: songbo63@126.com.

the existing technologies of ORM and providing uniform regulates and standards of data access. The architecture of JPA technology mainly includes two meta-data formats which are XML and JDK 5 annotation of JPA, JPQL (Java Persistence Query Language), JPA model structure and the API of JPA. This paper mainly focuses on analysis of the JPA structure and API.

JPA model shown in Fig. 1. In Java EE 6, the interface of JPA is included in two packages of javax.persistence and javax.persistence.spi. Most APIs of persistence package are annotations. Besides, it includes the persistence operation interface of Query and EntityManager. They are used to implement persistence operations interface. Four APIs in spi package are the service layer interface of JPA.

The interface of PersistenceProvider is provided by JPA manufacturer. It is called by initiator to create an EntityManagerFactory instance. PersistenceUnitInfo provides all information to create EntityManager instances. According to JPA specification, these information must store in the file of META-INF/ persistence.xml. By the addTransformer method of PersistenceUnitInfo interface, ClassTransformer interface is added. Also, ClassTransformer interface is put in container by PersistenceProvider interface. According to context parameters and information of PersistenceUnitInfo interface, EntityManager is created by EntityManagerFactory interface. In the light of PersistenceUnitInfo interface configuration, EntityManager interface creates all persistence entity. By createQuery method, the interface of Query is created. The interface with multiple methods to perform data querying. EntityTransaction interface provides transaction processing services.

From JPA model we can know that, the primary interfaces of JPA are the EntityManager and Query Interface. As the bridge of Object-Oriented Programming and Relational database, EntityManager explains specified ORM and puts the entity stored in database. It not only provides CRUD (creations, reads, updates and deletions) operations which are similar to SQL (Structured Query Language), but automatically maintains synchronization between the Entity and Relational database. EntityManager interface provides so many methods to create a search query of entity. Along with methods in Query interface, these methods provided by EntityManager interface implement actual query definition, parameter binding, performing and paging functions. Fig. 2 is the classical EAO patterns chart for the Course entity of e-Learning system.

Usually, the ORM framework in data persistence layer using DAO patterns to decouple data access code from business logic. Although the DAO (Data Access Object) and the EAO (Entity Access Object) are alteration of the same patterns, EAO patterns have been updated. They can be used in new DAO patterns of JPA (can be used in EJB 3 entity). Because entity is POJO (Plain Old Java Object), so there is no longer needed to Transfer Object/Data Transfer Object patterns while using EAO patterns.

Usually, each entity uses an EAO object. It is used for implement CRUD operation of entity. *CourseEAO* is the interface of EAO object. *CourseEAOImpl* is the implementation class of EAO. It is used for implementing business logic of *CourseEAO* entity. *CourseSessionBean* can uses *CourseEAO* and query the example of *CourseEntity*. It also can hide *CourseSessionBean*. According to EAO patterns, ORM framework has implemented decoupling with data persistence layer. But data access code cannot provide type-safe interface, it may be return incorrect type.

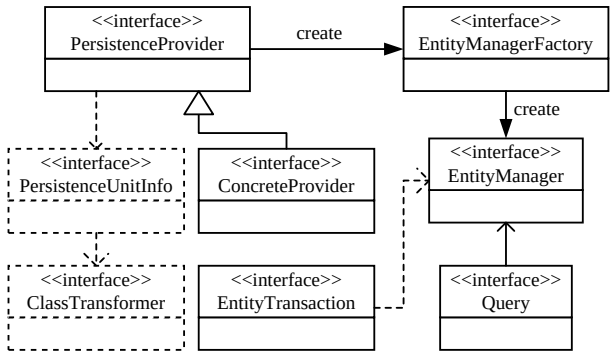

Figure. 1 JPA model.

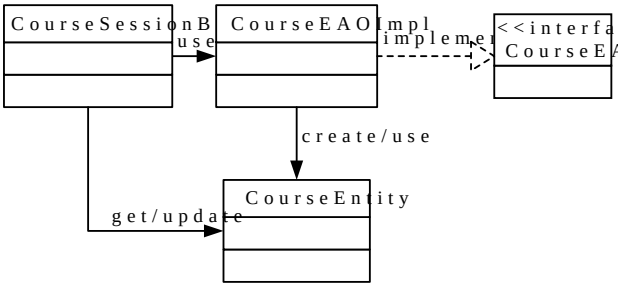

Figure. 2 EAO patterns.

Implementation of e-Learning system

The design of e-Learning system architecture follows the thinking of "programming face to interface", interface and their implementation consist in presentation layer, business logic layer and data persistence layer. In this way, the business layer uses the model of Façade [6] for presentation layer to provide a unified interface, and persistence layer uses the "generics EAO patterns" for the business layer to provide data abstract interface.

Design of generic EAO patterns. The design of generic EAO patterns is based on EJB 3 JPA model. Using Java generic technology [7], it makes model Interface simplified. Namely, it gives an abstract of the EAO and its implementation. The basic content of implementation is CRUD operation. It is shown in Fig. 3.

First we define a generic EAO interface with the methods we'd like all EAO's to share:

```
public interface IGenericEao<E, ID> {
    void persist(E entity);
    E findById(ID generatedId);
    void remove(E entity);
    public void saveOrUpdate(E entity);
}
```

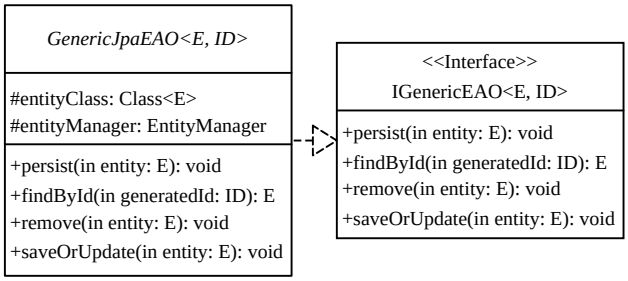

Figure. 3 generic EAO patterns.

The first type parameter, *K*, is the type to use as the key and the second type parameter, *E*, is the type of the entity. Next to the basic *persist*, *remove*, and *findById* methods, you might also like to add a List *findAll()* method. But like the entity class itself, we will revisit the EAO methods in later JPA implementation patterns. The second step is to create a base JPA EAO implementation. It will have basic implementation of all the methods in the standard EAO interface we created in step 1:

```
public abstract class GenericJpaEAO<E,ID> implements IGenericDao<E, ID> {
    protected Class<E> entityClass;
    @PersistenceContext protected EntityManager entityManager;
    public GenericJpaEAO( ) {
```

```
        ParameterizedType genericSuperclass=(Parameterize dType)
getClass().getGenericSuperclass();
        this.entityClass = (Class<E>)genericSuperclass.getAc tualTypeArguments()[1];
    }
    public void persist(E entity) {  entityManager.persist(entity);  }
    public void remove(E entity) {  entityManager.delete(entity);  }
    public E findById(ID id) {  return  entityManager.find(entityClass, id);  }
    public void saveOrUpdate(E entity) {  entityManager.saveOrUpdate(entity);  }
}
```

Firstly, in the definition of abstract class *GenericJpaEAO*, EntityManager can be obtained by @PersistenceContext annotation. It avoids importing EntityManager in all data access code and reduces the dependence of data access code [8]. Secondly, according to ParameterizedType, we can obtain Entity class of generic *E*. Finally, through the statement of entityClass and entityManager as protected type, they can be accessed by their sub-class directly.

Implementation of generic EAO patterns. EJB 3 JPA is introduced in data persistence layer, which is designed based on the model of Facade to define each module's business entrance of Facade interface in system, to define its implementation class, and to implement the generic EAO realization of specific business through the interface. The CRUD operation of the database is realized through generic EAO and the entity of corresponding module. The operation of data access is completed by JPA, and the system code itself does not directly access database. Take the following teachers teaching JPA implementation class of EAO in e-Learning system as an example to show the realization process, shown in Fig. 4.

First we define one subinterface for each entity type we want to persist, adding any entity specific methods we want. Teachers teaching EAO interface *ITeaCourseEAO* inherits common generics EAO interface, which is defined as follows:

```
public interface ITeaCourseEAO extends IGenericEAO < TeaCourse, String> {  }
```

Then we create such a specific EAO implementation. It extends the basic JPA EAO class and implements the specific EAO interface. *TeaCourseJpaDAO* which is the JPA implementation class of teachers teaching EAO realizes the teachers teaching EAO interface *ITeaCourseEAO*, and inherits the abstract class *GenericJpaEAO*. Thus, *TeaCourseJpaDAO* gets the standard methods on CRUD, which is defined as follows:

```
public class TeaCourseJpaEAO extends GenericJpaEAO <TeaCourse,String> implements
ITeaCourseEAO {  }
```

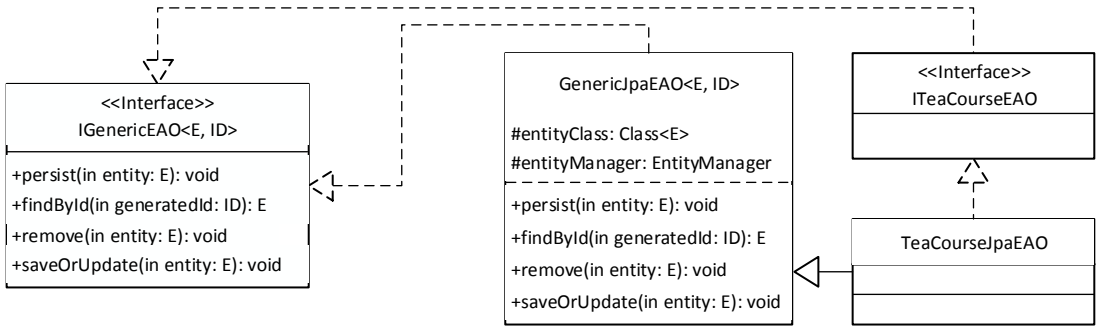

Figure. 4 Implementation of generic EAO patterns.

Generics definition explicitly specifies its type, which enhances the readability of the program. At the same time, types of inspections will be completed in the compiling phase to avoid the runtime exceptions, improve the operating efficiency. In spite of the basic data types may be used as parameter of type, and it will lead to boxing/unboxing cost, but it still have the advantages of enhancing the type

security and readability of the program. With this type-safe EAO pattern we get the following advantages:

- No direct dependency on the JPA API from client code.
- Type-safety through the use of generics. Any casts that still need to be done are handled in the EAO implementation.
- One logical place to group all entity-specific JPA code.
- One location to add transaction markers, debugging, profiling, etc. Although as we will see later, we will need to add transaction markers in other parts of our applications too.
- One class to test when testing the database access code. We will revisit this subject in a later JPA implementation pattern.

Conclusion

In this paper, it is simulated that the client-side submits static and dynamic request to Java EE application Server by browsers in real environment through the Grinder, an open source load generation and data collection tool [9]. The testing scenario is information query in the first page of system, which can relate to view switching, curriculum screening, sorting, and pagination etc. The performance testing highlights the Java class in the lasting unit, the ORM annotation indicated by the entity, and the database mapped by these Java classes. The aggregate average response time (AART) [10] of the system is shown as Fig. 5. When analyze the AART curve in Fig. 5, compared with the e-Learning system solutions [1] based on the open source framework Struts, Hibernate and Spring (SHS), we can find that the system response time of the solution proposed by this paper may increase according to the curve until it reaches 100 users. After that time, the increase of the curve changes more acutely. Through analyzing the changing tendency of curves, in the performance test, the application's maximum limit on the number of users is 100, so it meets the needs of the application service. In the platform of Java EE, the introduction of JPA specification marks the ORM persistence of object-oriented technology is maturing. After comparing with the e-Learning system solution of ORM framework, data persistence model of JPA provides a simple interface to JPA model manipulation by using the appearance model. Universal data access API which is designed by Java generic technology simplifies the development of object persistence. Putting generic EAO patterns in e-Learning system can greatly reduce the coupling degree; improve the portability of code and the speed of application development. Through the JPA and JSF, as well as UI components which are provided by ICEfaces, presentation layer of e-Learning system implements a loosely coupled, data-oriented RIA, while developing it in thin-client approach.

In the platform of Java EE, the introduction of JPA specification marks the ORM persistence of object-oriented technology is maturing. After comparing with the e-Learning system solution of ORM framework, data persistence model of JPA provides a simple interface to JPA model manipulation by using the appearance model. Universal data access API which is designed by Java generic technology simplifies the development of object persistence. Putting generic EAO patterns in e-Learning system can greatly reduce the coupling degree; improve the portability of code and the speed of application development. Through the JPA and JSF, as well as UI components which are provided by ICEfaces, presentation layer of e-Learning system implements a loosely coupled, data-oriented RIA, while developing it in thin-client approach.

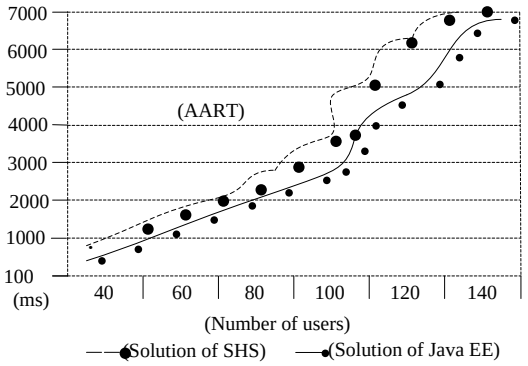

Figure. 5 AART curve.

References

[1] B. Song and J. Zhao, “Research on Network Teaching System Based on Open Source Framework”, IEEE Ninth International Conference on Hybrid Intelligent Systems, Vol.1, (2009), pp.28-32.

[2] B. Song and M. Du, “Lightweight Solution for J2EE Application and its Implementation”, Computer Engineering and Design, Vol.28, No.15, (2007), pp.3709-3712(in Chinese).

[3] Stephen M, “Rich Web Applications with Java and Ajax", http:// www.icefaces.org, 2005.

[4] ICEfaces, http://icefaces.org/main/downloads/os-downloads.ifaces.

[5] JSR-000220 Enterprise JavaBeans 3.0, http://jcp.org/aboutJava/comm unityprocess/final/jsr 220/index.html, (2008).

[6] Alur D, Crupi J, Malks D, “Core J2EE Patterns: Best Practices and Design Strategies”, American: Prentice Hall/Sun Microsystems Press, (2002), pp.185-198(in Chinese).

[7] Ye-Wang Chen, Jin-Shan Yu, “Generic Programming and Design Patterns”, Computer Science, (2006), Vol.33, No.4, (in Chinese).

[8] Panda D, Rahman R, Lane D, “EJB 3 in Action”, American: Posts & Telecom Press, (2008), pp.207-221(in Chinese).

[9] Grinder, http://grinder.sourceforge.net.

[10] Zadrozny P, Astom P, Osbome T,”J2EE Performance Testing With BEA Weblogic Server”, American: Expert Press, (2002), pp.9-17.