E-Learning System Development Using J2EE Framework Design Pattern

Zin Sandar Win, Sabai Phyu
University of Computer Studies, Yangon
zinsandarwin@gmail.com

Abstract

In the context of software engineering, the design patterns are reusable solutions to common problems and are one of the key mechanisms for implementing reliable and maintainable software. Design patterns are fundamental to design reuse in object-oriented development. E-learning offers a very flexible time and location independent way of learning for the participants. Therefore e-learning has advantages over traditional education and training such as reduced costs, improved consistency and timing, and so on. E-learning also includes Internet-based learning, web-based learning and online learning. Well-designed e-learning is effective giving learners an opportunity to practice their knowledge and skills. This paper describes development of e-learning system with J2EE design patterns using Struts, Spring, Hibernate framework and also utilizes MySQL database for data storage. This system aims for three types of users such as learners who are interested to learn and take examination about Information Technology (IT), coordinators to participate in teaching and administrator to manage the system.

Keywords: J2EE design patterns, Struts, Spring, Hibernate

1. Introduction

J2EE (Java 2 Platform, Enterprise Edition) is a component-based and platform-independent architecture for building enterprise applications. This architecture offers a multi-tiered distributed application model. Each tier is usually implemented by a different group of developers and communicates with the other tiers via a standardized interface. The advantages of n-tiered architecture are promoting software reusability, easier system maintenance and more effective use of data and networks. Most web-based enterprise applications are split into three logical tiers. The three logical tiers in n-tier architecture are: presentation tier, business tier (middle tier), and data tier (integration tier).

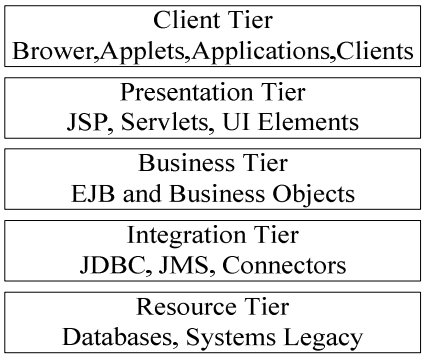

Figure 1. J2EE Pattern Catalog Address 3 Tiers

The J2EE pattern catalog currently includes 21 patterns [1].

Table 1. Patterns in the J2EE Pattern Catalog

Tier	Pattern Name
Presentation Tier	Intercepting Filter
	Front Controller
	Context Object
	Application Controller
	View Helper
	Composite View
	Service to Worker
	Dispatcher View
Business Tier	Business Delegate
	Service Locator
	Session Façade
	Application Service
	Business Object
	Composite Entity
	Transfer Object
	Transfer Object Assembler
	Value List Handler
Integration Tier	Data Access Object
	Service Activator
	Domain Store
	Web Service Broker

1.1. Presentation Tier

The presentation tier, which aims at presenting the business information to the user is implemented using

Servlets, JSPs and HTML/WML pages. Design patterns in presentation tier are:
Intercepting Filter: Facilitates pre-processing and post-processing of a request.
Front Controller: Provides a centralized controller for request handling.
Context Object: Encapsulates state in a protocol-independent way to be shared throughout your application.
Application Controller: Centralizes and modularizes action and view management.
View Helper: Separates processing logic from view.
Composite View: Creates a composite view from sub-views by separating content and layout management.
Service to Worker: Invokes business processing prior to view processing.
Dispatcher View: Invokes view processing prior to business.

1.2. Business Tier

The business tier (the middle tier), where core business mechanisms are implemented, is usually encapsulated in EJBs (Enterprise Java Beans). Design patterns in business tier are:
Business Delegate: Encapsulates access to a business service.
Service Locator: Centralizes lookup login for business services and components.
Session Façade: Exposes coarse-grained services to remote clients.
Application Service: Aggregates behaviour to provide a uniform service layer.
Business Object: Encapsulates business data and logic.
Composite Entity: Implements persistent business objects using entity beans.
Transfer Object: Carries data across a tier.
Transfer Object Assembler: Assembles a transfer object from multiple data sources.
Value List Handler: Manages search and iteration of a large set of results.

1.3. Integration Tier

The integration tier (the data tier), which represents different kinds of legacy systems, database servers, etc is usually accessed through the JDBC API and other standard interfaces provided by the J2EE Connector Architecture. Design patterns in integration tier are:
Data Access Object: Abstracts and encapsulates access to persistent store.
Service Activator: Provides asynchronous access to one or more services.
Domain Store: Provides transparent persistence for business objects.
Web Service Broker: Exposes one or more services using XML and web protocols.

1.4. J2EE Application Framework

J2EE Application Frameworks are the supporting technologies that maintain J2EE specification and work with J2EE. These are Struts, Tiles, Spring, Hibernate etc [4]. In this paper, Struts, Spring and Hibernate are used.
Struts: This is an open-source framework that in combination with standard Java technologies like Java Servlets, Java Beans, XML helps in effective development of web application. It is widely used in the development of application architecture based on the classic Model-View-Controller (MVC) design paradigm.
Spring: Spring framework is an open source application framework that solves many problems of J2EE. It is based on Java Bean configuration management with Inversion of Control principle (IoC). Its unique data access system with a simple JDBC framework improves its productivity with less error. Its Aspect Oriented Programming (AOP) program written in standard Java provides better transaction management services and also enables it for different applications. The main aim of Spring is to simplify the J2EE development and testing. Spring's goal is to be an entire application framework. Spring technology has features like Transaction Management, JDBC exception Handling, Integration with Hibernate, JDO, IBATIS, AOP framework, MVC framework.
Hibernate: It is a Java framework that provides object or relational mapping mechanism that helps in determining how Java Objects are stored and updated. It also offers query service or Java and helps in developing within the SQL and JDBC environment, and following some common Java idioms like inheritance, polymorphism, composition and collection. This kind of framework set up an easy way between the Java objects and the relational database.

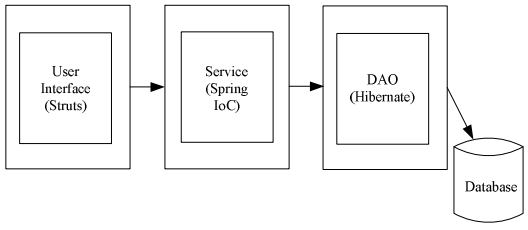

Figure 2. Struts, Spring, Hibernate Web Application

2. Related Work

There are several web applications using J2EE. J2EE provides standard solutions to each of the three logical tiers: Presentation Tier, Business Logic Tier

and Enterprise Information System Tier [3]. Many benefits of J2EE are preserved for Web Services such as Portability, Scalability, and Reliability. The integration of Web Services on the e-learning application domain using J2EE to integrate with Web service is depicted in [6]. A different elements regarding the requirements of an e-learning system, the users and their roles from the point of view the object oriented analyses and design methodologies are presented in [2]. J2EE technologies and design patterns are used to enable easy adding of new functions in web-based e-learning system [5]. Imed Hammouda, Kai Koskimies [3] showed how a general architectural tool called Fred ("Framework Editor") can be used to generate an architecture-centric task-based environment for developing J2EE applications. Mohammad Jahid Iqbal, Chandan Kumar Karmakar represented J2EE and its application framework for enterprise solutions [4].

3. Design Patterns used E-Learning System

In this session, we present four presentation tier patterns, three business tier patterns, one data tier pattern and entity bean and primary key patterns.

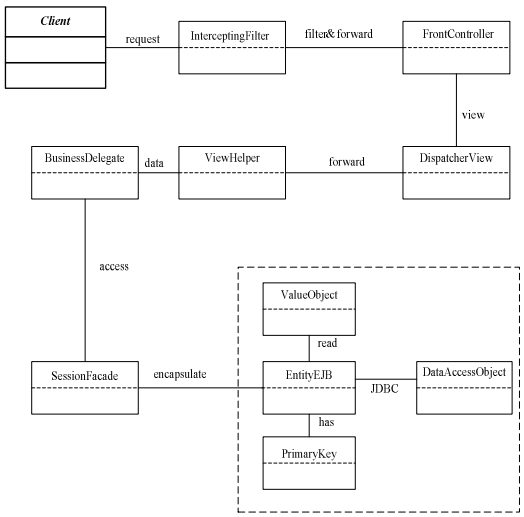

Figure 3. J2EE Pattern System

3.1 Intercepting Filter

Context: The presentation-tier request handling mechanism receives many different types of requests, which require varied types of processing. Some requests are simply forwarded to the appropriate handler component, while other requests must be modified, audited, or uncompressed before being further processed.
Problem: Preprocessing and post-processing of a client Web request and response are required.
Solution: Create pluggable filters to process common services in a standard manner without requiring changes to core request processing code. The filters intercept incoming requests and outgoing responses, allowing preprocessing and post-processing.
Usage: Login page is used to receive username and password.

3.2. Front Controller

Context: The presentation-tier request handling mechanism must control and coordinate processing of each user across multiple requests. Such control mechanisms may be managed in either a centralized or decentralized manner.
Problem: There is no centralized access point for presentation request handling.
Solution: Use a controller as the initial point of contact for handling a request. The controller manages the handling of the request, including invoking security services such as authentication and authorization, delegating business processing, managing the choice of an appropriate view, handling errors, and managing the selection of content creation strategies.
Usage: In Struts, the Controller is implemented by the ActionServlet class. The ActionServlet is declared in web.xml as follows:

```
<servlet>
    <servlet-name>action</servlet-name>
    <servlet-calss>
        org.apache.struts.action.ActionServlet
    </servlet-class>
</servlet>
```

All request URIs with the pattern *.do are mapped to this servlet in the deployment descriptor as follows:

```
<servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.do</url-pattern>
</serlvet-mapping>
```

3.3. Dispatcher View

Context: System controls flow of execution and access to presentation processing, which is responsible for generating dynamic content.
Problem: A view to handle a request and generate a response, while managing limited amounts of business processing.
Solution: Use Dispatcher View with views as the initial access point for a request. Business processing, if necessary in limited form, is managed by the views.
Usage: The behavior of the dispatcher, and the behavior of the request handlers that the dispatcher interacts with, is controlled via a configuration file struts-config.xml. For example, if username and password are correct, dispatch to appropriate welcome page. Otherwise, still in login page.

```
<form-beans>
     <form-bean name="userForm" type=
    "com.elearning.presentation.form.UserForm" />
</form-beans>
```

```
<action-mappings>
   <action path=”/login” validate=”true”
    type=”org.springframework.web.struts.
    DelegatingActionProxy” name=”userForm”
    scope=”session” input=”/home.jsp”>
        <forward name=”loginNoSuccess”
          path=”/home.jsp />
       <forward name=”loginLearnerSuccess”
         path=”/WEB-INF/jsp/welcome.jsp” />
       <forward name=”loginCoorSuccess”
         path=”/WEB-INF/jsp/welcomeCoor.jsp” />
       <forward name=”loginAdminSuccess”
         path=”WEB-INF/jsp/welcomeAdmin.jsp” />
  </action>
</action-mappings>
```

3.4. View Helper

Context: The system creates presentation content, which requires processing of dynamic business data.
Problem: To separate a view from its processing logic.
Solution: A view contains formatting code, delegating its processing responsibilities to its helper classes, implemented as JavaBeans or custom tags. Helpers also store the view's intermediate data model and serve as business data adapters.
Usage: For instance, the following code is used to show test result.

```
<logic:notEmpty: name=”marks”>
        Your score: ${marks} marks.
</logic:notEmpty>
```

3.5. Business Delegate

Context: A multi-tiered, distributed system requires remote method invocations to send and receive data across tiers. Clients are exposed to the complexity of dealing with distributed components.
Problem: To hide clients from the complexity of remote communication with business service components.
Solution: Use a Business Delegate to encapsulate access to a business service. The Business Delegate hides the implementation details of the business service, such as lookup and access mechanisms.
Usage: Action class calls methods in service class that abstracts and hides the implementation details of the business services.

3.6. Session Façade

Context: Enterprise beans encapsulate business logic and business data and expose their interfaces, and thus the complexity of the distributed services, to the client tier.
Problem: To expose business components and services to remote clients.
Solution: Use a Session Façade to encapsulate business-tier components and expose a coarse-grained service to remote clients. Clients access a Session Façade instead of accessing business components directly.
Usage: HttpSession is used such as

```
request.getSession().setAttribute(“userform”,
                                  myform);
```

3.7. Data Transfer Object (Value Object)

Context: Application clients need to exchange data with enterprise beans.
Problem: To transfer multiple data elements over a tier.
Solution: Use a Transfer Object to carry multiple data elements across a tier.
Usage: For example, user data transfer object (DTO) is used as a data carrier to save or update user information.

3.8. Data Access Object

Context: Access to data varies depending on the source of the data. Access to persistent storage, such as to a database, varies greatly depending on the type of storage (relational databases, object-oriented databases, flat files, and so forth) and the vendor implementation.
Problem: To encapsulate data access and manipulation in a separate layer.
Solution: Use a Data Access Object to abstract and encapsulate all access to the persistent store. The Data Access Object manages the connection with the data source to obtain and store data.
Usage: For instance, CourseDao interface and CourseDaoImpl class are used as data access mechanisms to access and manipulate data in course table.

3.9. Entity Bean

An *entity bean* represents a business object in a persistent storage mechanism. Entity beans are persistent, allow shared access, have primary keys, and may participate in relationships with other entity beans. Entity beans are user, course, coursecontent, and question.

3.10. Primary Key

Each entity bean has a unique object identifier. The unique identifier, or *primary key*, enables the client to locate a particular entity bean. For example, primary key pattern is used as getByCourseId (course_id) to get course in course content registration.

4. System Design and Implementation

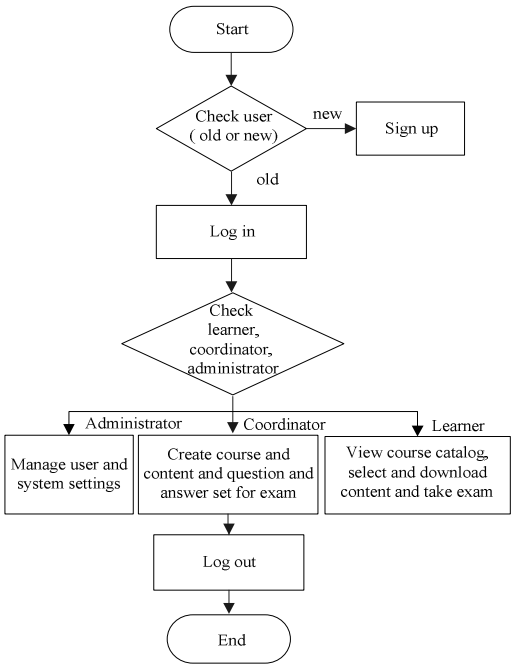

Figure 4. The Flowchart of System

In this system, there are three types of user level: Administrator, Coordinator and Learner. All users need to login with username and password. New user need to register for learner account. Coordinator and administrator accounts are registered by administrator. All users can update their own profiles. According to the login user types, their roles are different. Learner can recognize understanding of information technology (IT) by taking examination. Learner can also view course catalog, select and download course content to gain and improve knowledge. Coordinator can create course and course content about IT and also make question and answer sets for exam. Administrator can manage users and system settings. After using of the system, all users need to log out.

Intercepting Filters are used to pre-process and post-process of user requests such as logging and authentication. Front Controller manages the handling of the request, including invoking security services such as authentication and authorization, delegating business processing, managing the choice of an appropriate view. A controller coordinates with a dispatcher component. A dispatcher is responsible for view management and navigation. The user is dispatched to the next view using dispatcher components. View helpers are used to present business data in separate view components. Every generated entity bean has one primary key. If the entity bean has bean-managed persistence then a Data Access Object is used in order to encapsulate JDBC code. Instead of invoking multiple getters and setters for every field, a single method call is used to send and retrieve the Transfer Object/Value Objects. A number of Session Facades are used to encapsulate entity beans. Session Facade provide a simpler interface that reduces the number of business objects exposed to the client over the network and encapsulates the complexity of this interaction. Business Delegate hides the complexities of the services and acts as a simpler uniform interface to the business methods.

5. Conclusion

E-learning system describes the improvement of technology and high standard of technical education for learners. This system is intended to be able to use web-based learning and teaching system. In web-based e-learning system, learners can study lessons in 24 hours as they like, coordinators can participate in teaching and administrator can manage this system. By using J2EE technologies and design patterns, this system will be very easy for additional functionality to be added later.

6. References

[1] D. Alur, J. Crupi, and D. Malks, *Core J2EE Patterns: Best Practices and Design Strategies, Second Edition*, Prentice Hall PTR, June 10, 2003.

[2] I.A. Uta, “Developing E-learning System”.

[3] I. Hammouda, K. Koskimies, “A Pattern-Based J2EE Application Development Environment”.

[4] M.J. Iqbal, C.K. Karmakar, “J2EE and Its Application Framework for Enterprise Solutions”.

[5] M. Zdravkovic, M. Tranjanovic, and M. Ioannidis, “Functional Requirements Analysis and Design of J2EE Compliant Virtual Classroom Web Application”.

[6] X. Liu, A.E. Saddik, and N.D. Georganas, “An Implementable Architecture of an E-learning System”.