2014 3rd International Conference on Informatics, Environment, Energy and Applications
IPCBEE vol.66 (2014) © (2014) IACSIT Press, Singapore
DOI: 10.7763/IPCBEE. 2014. V66. 31

Research and Implementation on e-Learning System based on EGL

Xiaomei Li [1], Yan Xue [2] and Bo Song [3 +]

[1] Research and Training Center for Basic Education, Shenyang Normal University, Shenyang 110034, China
[2] Xinhua No.1 Primary School, Shuncheng District, Fushun City, Fushun 113006, China
[3] Software College, Shenyang Normal University, Shenyang 110034, China

Abstract. In the analysis of the relationship between EGL (Enterprise Generation Language) and Web 2.0 technologies, an application architecture of e-Learning system based on EGL and integrating Web 2.0 technology is proposed in this paper. Through the process of design and implementation of an e-Learning system, the key feature of the architecture is demonstrated – developers can focus on the business issues what code handle without caring for software technical details. The architecture is simple, easy to use and across languages, frameworks and runtime platforms, it not only can avoid the repeated writing the similar logic of each domain module, also is conducive to the robustness, maintainability of the system, and flexibility to the changes of whole business needs. The architecture can improve and perfect the traditional Java EE architecture. Utilizing the architecture in the application whose business logic is relatively complex has some guiding significance to the Java EE development.

Keywords: e-Learning, EGL, Web 2.0, architecture.

1. Introduction

With the development of Web 2.0 technology, the rich client in RIA (Rich Internet Application) is rapidly replacing the thin client in B/S. Because the e-Learning system can provide richer end-user experience, it has been adopted by more and more developers based on RIA architecture. For e-Learning system, the main benefit of Ajax is a greatly improved user experience. Although JavaScript and DHTML – the technical foundations of Ajax—have been available for years, most programmers ignored them because they were difficult to master. Although most of the Ajax frameworks available today simplify development work, you still need a good grasp of the technology stack. So, if you're planning to use Ajax to improve only your application's user experience – if you're not also using it as a strategic advantage for your business – it may be unwise to spend a lot of money and time on the technology [1]. ORM (Object-Relational Mapping) is a kind of technology which can solve the problem of impedance mismatch between Object-Oriented Programming and Relational Database. EJB 3, Hibernate and Oracle TopLink are the effective solutions to implement ORM [2] [3], but the implementation of ORM is time-consuming compared with JDBC [4]. Although the foregoing ORM tools provide convenience for operating Relational Database as object, the cost and complexity of e-Learning system are increased, and the real-time requirement of e-Learning system is reduced [5].

Aiming at the above-mentioned problems, an application architecture of e-Learning system based on EGL and Web 2.0 technology is proposed in this paper. Using the features of a cross platform and across application of EGL, the architecture can develop Web 2.0 application applied to browser-side and Java application running on server-side only with EGL. Developers do not need to master the two language – JavaScript and Java at the same time. The key feature of the architecture – developers can focus on the business issues what code handle without caring for software technical details – is implemented by using

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

existing platform and technology instead of replacing them and improve the real-time requirements and human- computer interaction experience of e-Learning system effectively.

2. EGL Overview

EGL is a high-level language that lets developers create business software without requiring that they have a detailed knowledge of runtime technologies or that they be familiar with object-oriented programming. The rational products that support EGL are based on Eclipse, which is the IDE (integrated development environment). EDT (EGL Development Tools) includes the core language packages – EGL SDK, and corresponding IDE [6]. EDT support the development of Web 2.0 application. Terminal user access the Web page (include HTML and JavaScript) generated by EGL code and the browser is responsible for download them to client-side. On the client-side, JavaScript generated by EGL code will interpreter in the browser and demonstrate the corresponding interface. Then JavaScript code generated by corresponding EGL statement is responsible for calling the Web Service or REST Service deployed on the server. Java EE container on the server-side is responsible for receiving the request from client-side and returning it to the browser, and then JavaScript application generated by EGL on the client-side will demonstrate it to the terminal user. EGL is not to replace existing technology and not to unify to develop new language and it is to maximize the use of mature technology as well as the supplement and extension of existing technology.

The EGL language is a full featured, procedural language that abstracts out the details of a target technology from developers. EGL has verbs like “get” that simplify the programming model by providing a consistent specification to a variety of target data sources. Writing your applications in EGL can also protect your development investment. The abstracted language can be cast or generated into any other language. As technology changes and evolves, your investment is protected by having the ability to re-generate into new target platform that have been improved or to entirely new platforms – without the need to modify your application. EGL has a construct called a Library. An EGL Library is simply a file that includes EGL code. EGL libraries provide the application developer with the ability to easily decouple the business logic from other application code. EGL Libraries provide a variety of entry points – one per function. These functions can be called from other functions in other Libraries or from EGL code in EGL Programs or EGL Page Handlers. EGL Programs can also be used to package up business logic, but with a single entry point. In an EGL based application, every page will have a “shadow” page handler. The EGL page handler controls a user's run-time interaction with a Web page. Specifically, the page handler provides data and services to the JSP that displays the page. It is considered a best practice that the page handler should have no logic. It implements the controller component of the MVC model. Although the page handler might include lightweight data validations such as range checks, it's best to invoke other programs or functions to perform complex business logic so that you follow MVC principles. Accessing data from databases can sometimes be challenging to developers whose primary objective is to provide their users with the information that is optimal for them to make business decisions.

3. Implementation of the e-Learning system

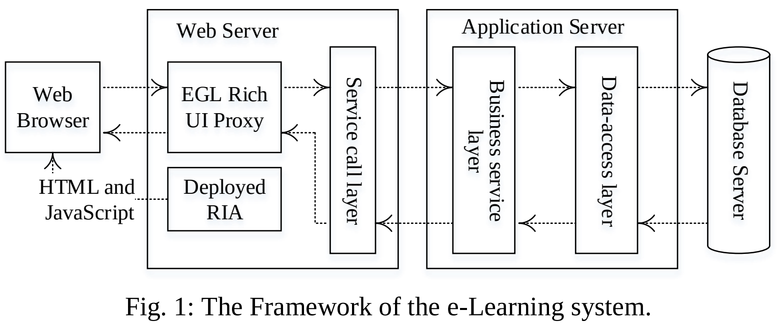

Fig. 1: The Framework of the e-Learning system.

As Fig.1 showed, a kind of hierarchical and extensible e-Learning system is proposed in this paper. After a Web server transmits the RUI (Rich User Interface) application to the user's browser, subsequent interaction with the server occurs only if the browser-based code accesses a service, which is a unit of logic that is more-or-less independent of any other unit of logic. The RUI application can access any number of discrete services. You automatically deploy the RUI application with the EGL Rich UI Proxy, which is EGL runtime code that handles the communication between the RUI application and the accessed services. The distinction among the tiers gives you a way to think about the different kinds of processing that occurs at run time. The architecture proposed by this paper makes full use of the characteristic of EGL technology and obtains further depuration and encapsulation, which enable the framework more suitable for the design and development of specific e-Learning system.

3.1. Database-Access Layer

The basic idea of a relational database is that data is stored in persistent tables. Each table column represents a discrete unit of data and each row represents a collection of such data and is equivalent to a file record. An EGL record can be the basic of a variable used as the source or target of an I/O operation [7]. We can interact with a relational database as follow: define a record whose stereotype is *SQLRecord*, create a variable based on that record and use the variable as an I/O object in different data-access statements. Usually, one or more columns in a database table can be primary keys, which mean that the values in those columns are unique to a given row.

3.2. Business Service Layer

Services can include new logic and can expose the data returned from other services and from called programs. EGL language offers end-to-end processing: developers can write the user interface, the service logic, and, if necessary, new backend programs. In many cases, before invoking the service, we need declare the variable based on the interface part in the client-side handler. Here is a declaration of *MyService*:

```
MyService SQLDataSource?;
dedicatedServiceBinding HTTPProxy;
```

In addition to the service variable declaration in the client-side handler, the service variable in the service is also needed.

```
MyService SQLDataSource? {@Resource{uri="binding:eLearnDerby"} };
```

The service variable declaration specifies the location which identifies the protocol that formats a message at the start of transmission and unformatted the message at the end of transmission. Web Service is a facility to let developers create logic that receives or sends messages over HTTP. The "*eLearnDerby*" is a connection to database. The called service can be available by binding the database connection. This information can all be included in a Web Services Description Language (WSDL) file [7].

3.3. Rich UI Interface

In the RUI design of e-Learning system, the basic design idea is to refresh RUI widget as a unit. That is, the whole page is divided into several widgets and each widget varies independently. When retrieving data by calling backend service, EDT need to refresh the front page and the grain size of refreshed widgets should be as small as possible, so that it can reduce the throughput and response time. In order to realize the design, a global access control point of the refresh widgets is needed. Because EGL access service using asynchronously calling, each calling will construct a callback function instance. The instance is responsible for refresh the user interface after returning the calling results. If the instance accesses the widgets of the page, it must have a reference to the widgets [7].

To create an EGL RUI application, a RUI handler is need primarily. The handler holds the EGL logic to add widgets to an initial DOM tree and to respond to events such as a user's click of a button. Primarily, an EGL RUI handler named MainHandler is created as a whole to call other handlers that can be designed as the sub modules of the e-Learning system by user's click of buttons. The buttons should be added an event named *showcall* as is shown below

```
onClick::= showcall;
```

Then we can configure the code so that the event handler responds to the event that is internal to the code. And the function *showcall* runs as soon as the user clicks the buttons. Such an event might be receipt of a message that was returned from a service. In the *MainHandler*, we can write the event handler and call other handlers by using a case statement. The code is shown below:

```
function showcall (event Event in)
    button DojoButton=event.widget;
    BoxContent.children = [ ];
    case (button.text)
      when ("HomePage") BoxContent.appendChild(new LoginHandler{}.ui);
      when ("e-Learn") BoxContent.appendChild(new Learning{}.ui);
      …
    end
end
```

The button widgets integrated the Dojo widgets and were placed into Box and GridLayout widget of the page. The page is divided into several sections by Layout widgets and the Box named *BoxContent* belongs to Layout widget. The sub modules will be the children of *BoxContent* and compose the single application by embedding multiple RUI handlers. However, by saying "embedded handlers" we do not mean to say that we physically embed one handler in another. Instead, one handler – an EGL part that present the user interface – declares a variable used to access the functions and widgets in a second handler. For example, we can not only declare a variable that provides access to the handler *LoginHandler* as shown below, but also access it directly using "*new*" keyword as shown in the case statement of the function *showcall*.

```
myLoginHandler LoginHandler{ };
```

A reasonable practice is to use embedded handlers for service invocation and for other business processing that lacks a user interface. If the embedded handler has an on- construction function, the function runs when the declaration for the related variable runs.

The core module of the e-Learning system is Learning Module which is included in the hander named *Learning {}*. It encapsulates the references of some RUI widgets such as buttons including events, dataGrid displaying the information of lessons of the e-Learning system from database. These events can be executed by calling service from business service layer. Each service calling will construct a callback function instance and retrieve the references of the needed RUI widgets.

By the RUI, service calling layer can be responsible for calling the service provided by the Business service layer on the server-side and then return the results to client logic layer and client presentation. That is to say, service calling layer decouple the front logic from the backend logic and use call statement to call the created *MyService* of each service. The access of Backend service becomes simpler and the code becomes easier to maintain.

Then Take function *readFromTable* for example to demonstrate the implementation principle of service calling layer. The widgets of client-side include UI controls and each widget can indicate screen events such as "*onClick*" to call the code of service layer.

```
function readFromTable (event Event in)
    call MyService.getAllLessons( )  using dedicatedServiceBinding returning to mycallback
    onException serviceExceptionHandler;
end
```

The function *readFromTable* is responsible for retrieving data and displaying the data. As is show in the code of the function, the screen event "*onClick*" will call the service of *MyService* using dedicatedServiceBinding. Resource Binding is one of the outstanding characteristics of EGL language. This simply means it is a description about how to connect to the database and how to invoke the service. We can maintain the binding in the deployment descriptor files of EGL and the binding can be seen the extending of the application logic. When we develop and deploy the application, the deployment descriptor files will provide specific details of connection and calling service. If errors occur in the process of calling service, exception handler *serviceExceptionHandler* will be called.

After the screen event "*onClick*" retrieve the data from the database by service calling layer, and then it will use the function *mycallback* to process it. In this case, the DataGrid widget named *myLesson* is responsible for displaying the data. The code of *mycallback* is shown as below:

```
function mycallback (retResult Lesson[] in)
    myLesson = retResult;
    myLesson_ui.data = myLesson as any [];
end
```

We can access a function or property in an embedded widget by extending the dot syntax. For example, the above-mentioned statement retrieves the displayed data of the DataGrid widget named *myLesson*.

In the Learning handler, users can not only retrieve the information of lessons from database, but also add, delete or edit the lessons into the RUI. These events can also be realized by buttons and they can be displayed by function *showDialog* and also be hided by function *hideDialog*.

```
function showDialog (event Event in)
    Dialogcontent.children = [info,buttonBar];
   dialog.showDialog();
end
function hideDialog (event Event in)
    dialog.hideDialog();
end
```

As is shown in Fig.2, the function readFromTable creates the screen event "OnClick" and prepares to call the service with call statement. The function getAllLessons is a function of the service which invokes access the database with SQL statement. The function onException is responsible for an error value in the process of data access. The function updateAll is a callback function and is responsible for handling the data returned from the service. The function readFromTable and updateAll are all in the handler of client-side as Rich UI and the function getAllLessons in the service file locate in business service layer in server-side. The focus of business service logic layer is the development of business rules and the implementation of business flow which is related to business requirements.

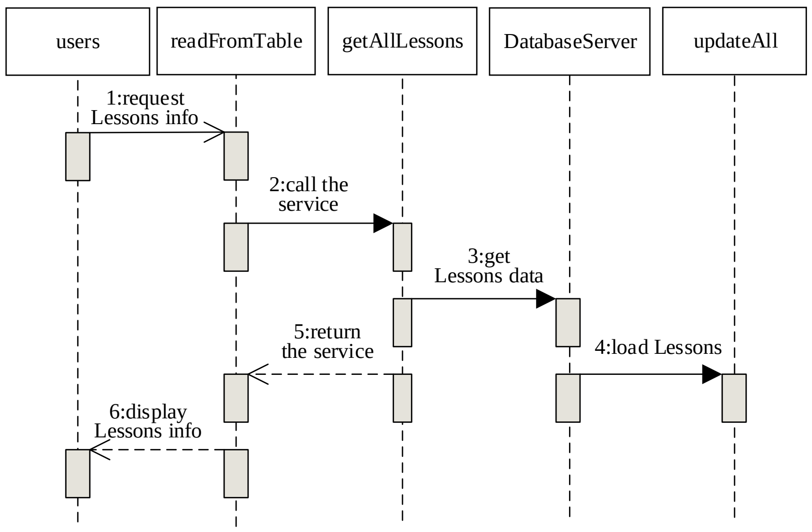

Fig. 2: The call flows between client-side and server-side.

Any technique for working with a service is a variation on what we have shown here: create a variable, bind it to a service client binding, and access a function by way of the variable. Remote Services are the standard way to communicate with the Web application server from the user's browser. This is done using a standard Ajax, essentially an HTTP POST. The server-side code for that Ajax call is found within the generated JavaScript application. This is code that executes a Web application server and this is where we tie user actions at the browser into business logic and then for data persistence storage.

4. Conclusion

In this paper, the architecture is designed for ease of use and eliminate common, and often taken for granted, overhead in all stages of application development. The primary benefit of the application architecture of e-Learning system based on EGL is simplicity. The separation of Model and View also allows for a division of labor. For example, one developer might handle database access and business logic manipulation while another developer may focus on the development of user interface. This division lets developers fulfill a task appropriate to their profession and lets different tasks proceed in parallel.

Finally, the application architecture proposed in this paper is based on EGL and improves the development efficiency of e-Learning system. It will have an important guiding significance for the application and development of the e-Learning system in the network education.

5. Acknowledgements

This work was supported by the Science and Technology Project of Education Department of Liaoning Province, China (①.Research of e-Learning System of small and medium-size Enterprises Based on SaaS, No.L2013417. ②.Research on strategy for balanced development of elementary education, No.JG12CB215).

6. References

[1] B. Song and M. Y. Li, “An e-Learning System Based on GWT and Berkeley DB,” Lecture Notes in Computer Science 7332, Vol.2, pp. 26-32, 2012.

[2] B. Song and J. Liu, “Implementation of J2EE Data Persistence Tier with TopLink,” Microelectronics & Computer, 23(8), pp. 132-135, 2006.

[3] 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, IEEE: Shenyang, pp. 28-32, 2009.

[4] W. Fang and Y. Sun, “Research and Application of J2EE's Data Persistence Layer”, Computer Technology and Development, 17(2), pp. 68-91, 2007.

[5] B. Song and Y. Zhang, “Implementation on Network Teaching System Based on Java EE Architecture”, IEEE Second International Conference on Information Technology and Computer Science, Vol.1, Kiev, pp. 354-357, 2010.

[6] IBM Corporation, “EGL Programmer's Guide Version7 Release 00,” USA, 2007.

[7] B. Margolis and Danny, “Enterprise Web 2.0 with EGL,” MC Press: USA, pp. 225-237, pp. 83-95, pp. 131-141, 2009.