An e-Learning System Based on EGL and Web 2.0

Bo Song and Jie Gao

***Abstract*—Aiming at the existing problems of e-Learning system application architecture, in the analysis of the relationship between EGL 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. In addition, it can reduce the cost during the development stage of application and effectively improve the real-time requirements and human-computer interaction experience of e-Learning system.**

***Index Terms*—E-Learning, EGL, Web 2.0, architecture.**

I. 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 [1]. 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 [2]. ORM (Object-Relational Mapping) is a kind of technology which can solve the problem of impedance mismatch between Object-Oriented Programming and RDB (Relational Database). EJB 3, Hibernate and Oracle TopLink are the effective solutions to implement ORM [3], [4], but the implementation of ORM is time-consuming compared with JDBC [5]. Although the foregoing ORM tools provide convenience for operating RDB as object, the cost and complexity of e-Learning system are increased, and the real-time requirement of e-Learning system is reduced [6].

Aiming at the above-mentioned problems, an application architecture of e-Learning system based on EGL (Enterprise Generation Language) 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 languages – 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 existing platform and technology instead of replacing them and improve the real-time requirements and human-computer interaction experience of e-Learning system effectively.

Manuscript received March 5, 2014; revised May 15, 2014. This work was supported by the Science and Technology Project of Education Department of Liaoning Province, China (Research of e-Learning System of small and medium-size Enterprises Based on SaaS, No. L2013417).

Bo Song and Jie Gao are with Software College, Shenyang Normal University, Shenyang, Liaoning, China (e-mail: songbo63@aliyun.com, gaojiexy@126.com).

II. EGL and Web 2.0

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 language is architected to reflect patterns that are common to different kinds of business software, and the language hides many details that are platform specific. EGL also helps a company retain developers who are knowledgeable in business processes, even if those developers lack the time needed to stay current with technical change. And the relative simplicity of the language helps traditional developers become accustomed to the latest technologies.

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 [7]. In the EDT environment, EGL is used in a development process that has defined steps, from coding a source to generating an output (Java, or JavaScript) to preparing and deploying that output. The first stage is to compile the EGL code and EGL compiler will scan and analyze the syntax and semantic of EGL source file, then generate the IR (intermediate representation). The second stage is code generation. The code generator read and parses the IR (eglxml file) and then generates corresponding target language. The third stage is to use target language compiler to compile into executable code. The Java compiler will compile the generated Java source files into class files to run in different platforms and the generated JavaScript code will be interpreted in the browser environment.

EDT support the development of Web 2.0 application and its deployment. 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.

As is shown in the compilation process of EGL, EGL itself does not run directly and it will be compiled and executed by the compiler of target language in building and generating target language on corresponding platform. And in the process of language generation, several of different platforms and existing technology can be integrated and made full use of. For example, in the browser-side, EDT made full use of Dojo framework to support the development of Web 2.0 and JavaScript generated by EGL encapsulated the Dojo [8]; in the server-side, Java code generated by EGL encapsulated database access using JDBC. 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.

III. Application Architecture with EGL

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 RUI 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.

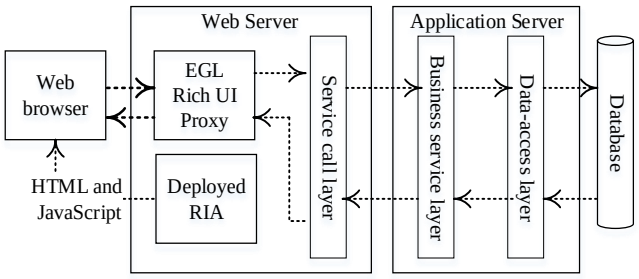

Fig. 1. The architecture of the e-learning system.

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. The EGL RUI Proxy is runtime software that is installed with your RUI application if the deployment target is IBM WebSphere Application Server or Apache Tomcat. The EGL RUI Proxy handles communication between the application and any services that are accessed by the application [9].

The RUI application uses the EGL RUI Proxy to access every invoked service, even services that are on the same server. A tier is logical in the sense that some or all can be on the same machine. The distinction among the tiers gives you a way to think about the different kinds of processing that occurs at run time. After a Web server transmits the RUI 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. A service might be half a world away from the user. The RUI application can access any number of discrete services, and each might do a simple task such as provide details on a property. However, enterprise development often involves access of service-oriented applications. Each of these applications is composed of services that work as a unit to fulfill a specific and complex purpose. You automatically deploy the RUI application with the EGL RUI Proxy, which is EGL runtime code that handles the communication between the RUI application and the accessed services.

IV. Implementation of the e-Learning System

A. 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.

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
```

B. 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.

A RUI application invokes services asynchronously, which means that the user can still interact with the user interface while the RUI application is waiting for the service to respond. However, if the user needs the information to continue a task, we can disable widgets and present a simple animation until the service responds. The runtime technology ensures that the invocation occurs as soon as a message arrives from the service. The process for invoking a service often requires an EGL interface part, which describes the data that can pass between the application and service. The Interface part tells the names of the service operations and, for a given operation, the kinds of data that the application exchanges with the service [9].

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. A service contains public functions that can be accessed from other code and can include private functions and global variables, but those functions and variables are solely for use by functions that are within the service. At run time, the service is stateless, which means that the internal logic never relies on data from a previous invocation. For example, the collaboration diagram (Fig. 2) below presents a typical flow of calls between client-side and server-side.

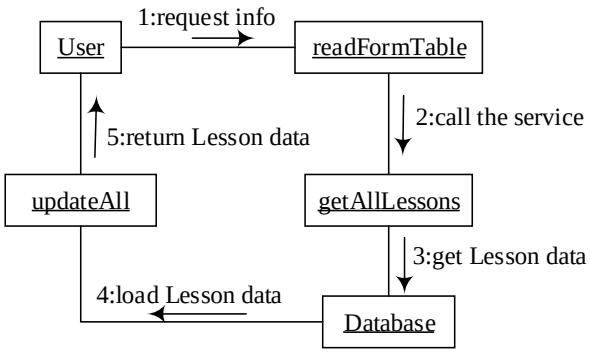

Fig. 2. The call flows between client-side and server-side.

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 RUI 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.

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.

C. 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 [9]. 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. For example, here is a Record about lessons of the e-Learning system:

```
record Lesson type Entity {
@table{name = "USER.LESSON"}}
lesson_id int { @id, @GeneratedValue,
@Column{name="lesson_id"}};
lessonname string(100) {
@Column{name="lessonname"}};
lessonintro string(255){
@Column{name="lessoninfo"}};
end
```

The *lesson_id* column is an identity column, which means that the database will place a unique value into that column whenever the user creates a record. Each value is one more than the last. To generate the Java code that is appropriate for the SQL operation, the EDT Java generator uses the Table, ID, Generated Value, and Column annotations.

Access to relational database is by way of SQL. To retrieve a row, we can assign a value to the record field associated with the key column and then issue a get SQL statement to read data from the table in database. The code of function *getAllLessons* using get SQL statement is shown below:

```
function getAllLessons () returns (Lesson [])
Lessons Lesson[];
try get lessons from MyService with #sql { select  lesson_id,
lessonname, lessonintro from LESSON};
onException (ex sqlException)
end
return (Lessons);
end
```

The following code declares a related record variable in the client-side handler. And then we can use the variable to access the database.

```
myLesson Lesson [ ];
```

In addition, before we generate EGL code, we should configure a build descriptor, which is a build part that guides the generation process and references other definitions. The resource associations can associate the logical file name with a physical file on each target platform where we intend to run the code.

V. Conclusion

In this paper, it is simulated that the client-side submits request to the Web server-side in the EDT development environment to propose application architecture of e-Learning system based on EGL and Web 2.0. The testing scenario is information query in the handler of client-side, users can create and connect the database connection and retrieve the data in database through access the service. The aggregate average response time (AART) of the system is shown as Fig. 3.

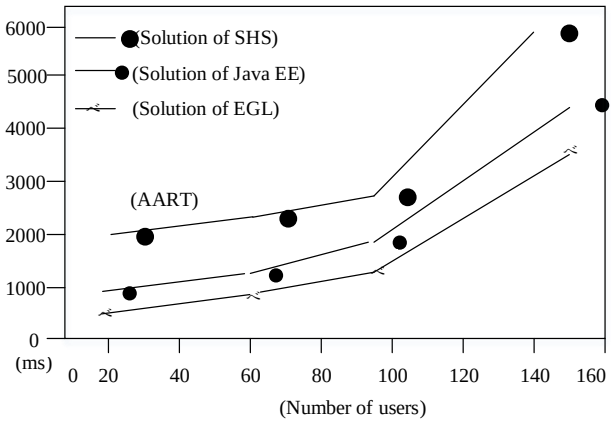

Fig. 3. AART curve.

Analysis of AART curve in Fig. 3, compared with the e-Learning system solution based on open source framework — Struts, Hibernate and Spring (SHS) [4], the system response time of the solution of Java EE increases in accordance with the linear way until it reaches about 100 users [6]. After that time, the increase of the curve becomes more intense. By analyzing the change trend of the two curves, we can draw the conclusion: the maximum number of users of the solution of Java EE presents is 100 and the application is suitable for small e-Learning system. However, the solution can significantly reduce the cost and complexity of e-Learning system and improve the e-Learning system interactive experience and real time requirements. Because EGL language can be executed after generating Java code, the changes of the curve of the solution of EGL are similar to Java EE's and only its speed is slightly slow. It implements the loose coupling between business logic and access control, that is, choices related to a user interface, and choices related to the data in persistent, are handled separately. EGL is general because the technologies are so varied, but we will follow up by nothing how the separation applies to EGL Rich UI. In the development of EGL Rich UI application, the MVC pattern is often used to realize above-mentioned features. Therefore, 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. 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 improves the development efficiency of e-Learning system. 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. It will have an important guiding significance for the application and development of the e-Learning system in the network education.

Acknowledgment

This work was supported by the Science and Technology Project of Education Department of Liaoning Province, China (Research of e-Learning System of small and medium-size Enterprises Based on SaaS, No.L2013417).

References

[1] N. Goel, S. Maitrey, and S. Kanauzuya, "Web technologies (Web2.0)," *International Journal of Computer Applications*, vol. 1, no. 6, pp. 11-12, 2010.
[2] 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.
[3] B. Song and J. Liu, "Implementation of J2EE data persistence tier with TopLink," *Microelectronics & Computer*, vol. 23, no. 8, pp. 132-135, 2006.
[4] B. Song and J. Zhao, "Research on network teaching system based on open source framework," in *Proc. IEEE Ninth International Conference on Hybrid Intelligent Systems*, 2009, vol. 1, pp. 28-32.
[5] W. Fang and Y. Sun, "Research and application of J2EE's data persistence layer," *Computer Technology and Development*, vol. 17, no. 2, pp. 68-91, 2007.
[6] B. Song and Y. Zhang, "Implementation on network teaching system based on Java EE architecture," in *Proc. IEEE Second International Conference on Information Technology and Computer Science*, Kiev, vol. 1, 2010, pp. 354-357.
[7] *EGL Programmer's Guide Version7 Release 00*, IBM Corporation, USA, 2007.
[8] M. A. Russell, *DOJO: The Definitive Guide*, O'Reilly Media: USA, 2009.
[9] B. Margolis and Danny, *Enterprise Web 2.0 with EGL*, MC Press: USA, 2009, pp. 83-95.

Bo Song is a full professor of the Software College of the Shenyang Normal University. He received the M.S. degree from the University of Fukuoka Education, Japan in 1999. His main topics of interest are software engineering, e−learning, collective intelligence.

Jie Gao is a postgraduate of the Software College of the Shenyang Normal University. She received the B.S. degree from the Yantai Nanshan University, Yantai city, Shandong Province, in 2012. Now she is studying for a master's degree in the Shenyang Normal University and will acquire it in 2015. Her main topics of interest are software engineering, e−learning, collective intelligence.