ISBN 978-952-5726-11-4
Proceedings of the Third International Symposium on Electronic Commerce and Security Workshops(ISECS '10)
Guangzhou, P. R. China, 29-31,July 2010, pp. 231-234

Rapid Development of Web Applications Based on SmartGWT

Dongjin Yu[1], and Pei Zhang[2]
School of Computer, Hangzhou Dianzi University, Hangzhou, China
[1]Email: yudj@hdu.edu.cn
[2]Email: zhang_pei880702@yahoo.com.cn

***Abstract*—With the popularity of Web development patterns and diversification of business requirements, development of Web applications based on frameworks has showed remarkable advantages such as simplifying development processes and improving efficiency of software development. However, traditional Web-based frameworks usually do not care the consistency and interactivity of their user interfaces. This paper presents the approach to rapid development of Web applications using SmartGWT, a GWT-based framework, as well as Ajax technologies. It also gives the real case applied to the development of Role Based Access Control System, which proves that the approach could accelerate the development process of Web-based applications.**

***Index Terms*—Web Framework, Google Web Toolkit, SmartGWT, Rapid Application Development**

I. INTRODUCTION

With the wide spread of Rich Internet Applications (RIAs), using Ajax techniques to build Web applications has caused significant changes in Web-based development processes. From the first generation of Web frameworks represented by Apache Struts, Web application frameworks on Java have got sufficiently developed. Moreover, many old-fashioned Web applications are being replaced by RIAs [1], which provides a new client-server architecture that avoids the latency of roundtrips to the server by processing locally on the client. There are many features which characterize RIA from a Web Engineering point of view such as rich interaction capabilities, complex client-side processing, the elimination of full page refreshing to provide navigation, multimedia animation, etc [2]. However, RIAs are usually complex applications and their development requires designing and implementation which are time-consuming and error-prone [3].

Fortunately, since the release of Google Web Toolkit (GWT) and related RIA widget frameworks such as SmartGWT, RIA development is now becoming easier and more flexible. SmartGWT is a GWT-based framework that allows developers to not only utilize its comprehensive widget library for user interfaces, but also tie these widgets in with server-side for data management. As a perfect implementation of RIA, SmartGWT introduces new structural and behavioral models in order to help developers construct Web applications that look and function like traditional desktop applications.

Software architecture involves the structure and organization by which system components and subsystems interact to form systems and the properties of systems that can best be designed and analyzed at the system level [4]. As a development toolkit for optimizing browser-based applications, GWT framework just focuses on presentation layer and business logic layer, not for all layers which Web application usually have. Based on the study of GWT features, this paper introduces a layered and extensible Rapid Application Development Framework (RADF), combined with SmartGWT, which contains a wealth of enterprise-grade interactive widgets, layouts and data sources. In order to avoid the possible limitations of widgets provided of SmartGWT, RADF incorporates many extensions to enhance the function of SmartGWT's widget library.

This paper is organized in the following manner. Section 2 introduces the overall architecture and its hierarchical structure of RADF. Section 3, Section 4 and Section 5 give the design of its presentation layer, business logic layer and data persistence layer respectively. After showing a real case based on RADF in Section 6, final section concludes this paper and discusses the future research directions.

II. ARCHITECTURE OF RADF

Over the past few years, a large body of research and development covers integration at the data and business logic layer, but few investigations examine it at the presentation layer [5]. Instead, RADF based on SmartGWT adopts a typically multi-layer structure, allowing easier implementation of Rich Internet Applications. Moreover, each layer in RADF is independent and loosely coupled with the others.

RADF is divided into three layers: layer of user interface, layer of process control and layer of business logic, allowing developers of different roles to chase higher efficiency by means of parallel working. In particular, to reduce the development complexity of user interface based on GWT, RADF presents Graphical User Interface (GUI) widgets and layout frameworks, which facilitates the development of user interface layout. In addition, RADF establishes a fine-grained interactive process between the client and the server layers. Fig. 1 shows the graphical representation of the RADF-based application architecture and features of the overall skeleton.

III. LAYER OF PRESENTATION

Since RIA has a rich interactive user interface similar to desktop application, developers must complete the static features of widgets with a model that will allow specifying the interaction between these widgets and the rest of the system [3]. RADF makes full use of widget library provided by SmartGWT and meantime extends their functionalities. Each encapsulated widget is rich with configurable options to perform various display effect and behavior.

Different from traditional frameworks based on MVC design pattern, RADF reduces significantly network traffic using more intelligent asynchronous request that sends only small blocks of data. In other words, more processing is done on the view layer, instead of the server side. Through the extension of widget library provided by SmartGWT, RADF provides efficient management of local data in ways that will truly bring applications perfect user-interface by quickly and effectively redisplaying data shown previously. On the other hand, making full use of the processing capability of client-side can improve the response of RADF-based applications and reduce the load of server-side. During this process, DataSource Object plays a key role in transmitting data between client side and server side. Before calling a RemoteService, the widget containing UI controls must be bound with a certain DataSource Object with the local copy of data that components wish to display or manipulate. An AsynCallBack method is then created and passed to the server side. The ModelDriven action object invoked by this method will eventually create a model object representing the data delivered from the browser to the server and then handled by the logic layer via calls through Business Process Objects, or simply BPOs. Using data objects like BeanModel and DataSource, developers are able to take advantage of the most advanced and extendable data widget, such as Grid, Tree and ListView.

IV. Layer of Business Logic

The business logic layer introduces the well-known Web architecture called Representational State Transfer (REST), embracing a stateless client-server architecture in which Web services are viewed as resources and can be identified by their URLs. The HTTP methods such as GET and POST are the verbs that the developer can use to describe the necessary create, read, update, and delete (CRUD) actions to be performed [6], using a globally defined set of remote methods to process client-side request.

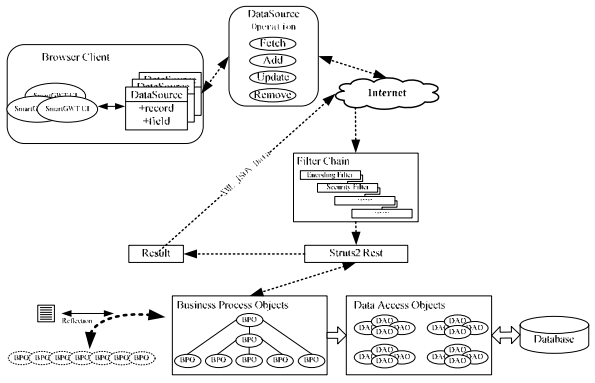

Figure 1. Architecture of RADF-based application.

RADF encapsulate the handling processes of requested information into Business Process Objects (BPOs). When RADF-based applications boot up, BPOs are automatically loaded and configured in Servlet Context, and then prepared to be used in the business logic layer if requested in future. Once the request is sent to server-side, the model-driven action class, similar to POJO, performs the interaction with BPOs via Data Transfer Objects (DTOs) to transform business operations into basic database operations. Different from traditional entity objects, these DTOs, containing only basic data types and operations, must implement the GWT IsSerializable interface.

Fig. 2 shows class templates of business logic layer, which decouple between foreground logic and background logic. Moreover, each class of business logic layer and configuration file has the predefined templates to be extended or filled.

V. Layer of Data Persistence

Business Process Objects accomplish access to database by Data Access Objects (DAOs). When the business logic layer transfers business logics to the data persistence layer in the type of SQL statements and entity objects, the DAO supporting classes, containing basic methods of data inserting, updating, deleting and querying, will fulfill specific data manipulations.

The statements of dynamic SQL templates used for data manipulation could be configured in property files. They are automatically loaded to Servlet Context when the application boots up. The Data Persistence Layer is responsible for mapping input and output parameters to object-oriented entities. In other words, the property of Data Transfer Objet, transmitted from business logic layer, is mapped to the parameter list of SQL statements, which is implemented via reflected mechanism and then executed to realize the access to database. Besides, the Data Persistence Layer returns the result set taken from database to business logic layer as an entity object.

VI. Case Studies

RADF focuses on building a series of reusable software modules and provides many basic services. It predefines the structure of application software to meet domain requirements, such as UI composition framework, data source framework, and business process framework. The following gives the real case of generating business data sources in the development of Role Based Access Control System using RADF. DataSource Module plays a key role in implementing the asynchronous transfer of information, bridging the UI composition and server service.

In the case, BaseDataSource is the base class that application extends to access to business data sources by using RADF which inherits core operation methods of RestDataSource, such as fetch, update, create and remove, and encapsulates the configurations for paging. Taken the query of role information for example, the data source of role information would define data source fields corresponding to columns of table in the database and set the properties to call the background service in its construction method.

```
public class RolesmanagerDS extends BaseDataSource {
    public static RolesmanagerDS getInstance() {
        if (instance == null) {
            instance = new RolesmanagerDS("RolesmanagerDS ");
        }
        return instance;
    }
    public RolesmanagerDS (String id) {
        setDataURL("rolesmanager.xml");
        //define fields of datasource according to properties of model
        DataSourceField aae017Td = new DataSourceField(
            "aae017", FieldType.TEXT, "RoleName",20);
        ……
        OperationBinding fetch = new OperationBinding();
        // telling a DataSource how to execute basic DS operations
        fetch.setOperationType(DSOperationType.FETCH);
        fetch.setDataProtocol(DSProtocol.GETPARAMS);
        ……
    }
}
```

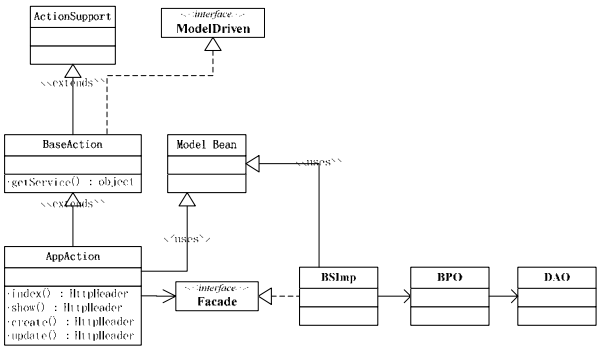

Figure 2. Class templates of business logic layer.

UI widgets, such as list grids, could be then bounded with the DataSource Object as the following manner.

```
ListGrid roleListGrid = new ListGrid();
roleListGrid.setDataSource(RolesmanagerDS.getInstance());
```

Here, ListGrid is a DataBoundComponent representing a list of objects in a grid. Meanwhile, RolesmanagerDS would send an asynchronous request, if necessary, to RolesmanagerAction extended from BaseAction according to the dataurl property of RolesmanagerDS. Here, BaseAction, as the super class of all Actions, is roughly defined as following.

```
public class BaseAction extends ActionSupport implements
ModelDriven<Object>, Validateable {
    private Object model = new Object(); //request data
    protected ActionListModel listModel; //response data
    public Object getService(String facade)throws AppException {
        return FacadeFactory.getService(facade);
    }
    public Object getModel() {
        return listModel.getlist(); //return the response data
    }
}
```

In this way, ListGrid would eventually display related data stored in data source. As shown in Fig. 3, after clicking the button of ShowMenuOptions, Section B would synchronously display related menu options

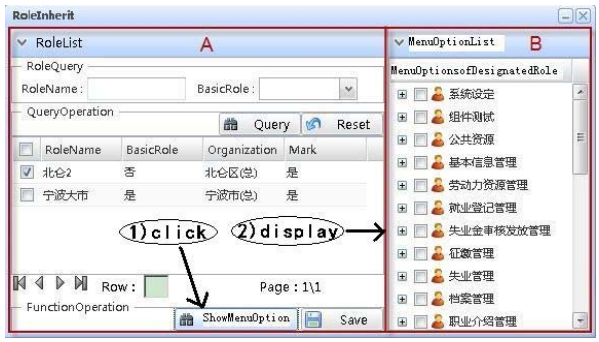

Figure 3. Screenshot of query of menu options.

granted to the designated role in Section A.

VII. Conclusions

SmartGWT delivers all the power of true AJAX, which greatly accelerates the integration with existing Java business logic and custom data tiers. This paper focuses on infusing the Rapid Development Application Framework with SmartGWT to help developers flexibly develop AJAX Web-based applications. Since the update of SmartGWT framework is fast and new widgets emerge continually, RADF on SmartGWT should keep updated to keep up with SmartGWT.

Acknowledgment

The work is supported by the Foundation of Key Science and Technology Projects (No. 2008C11099-1), China Innovation Foundation for Technology-based Firms (No. 08C26213300677), and the Science Foundation of the Hangzhou Dianzi University (No. KYS055608069).

References

[1] Driver M, Valdes R, and Phifer G., “Rich Internet Applications are the next evolution of the Web”, *Technical Report*, Gartner, 2005.

[2] Meliá S., Gomez J., “The WebSA Approach: Applying Model Driven Engineering to Web Applications”, *Journal of Web Engineering*, Vol. 5, No. 2, pp. 121-149, 2006.

[3] Santiago Meliá, and Jaime Gómez, “A Model-Driven Development for GWT-Based Rich Internet Applications with OOH4RIA”, *Eighth International Conference on Web Engineering*, pp. 13-23, 2008.

[4] Philippe Kruchten, Henk Obbink, and Judith Stafford, “The Past, Present, and Future of Software Architecture”, *IEEE Software*, pp. 22-30, March/April (2006).

[5] Linaje M., Preciado J. C., and Sánchez-Figueroa F., “Engineering Rich Internet Application User Interfaces over Legacy Web Models”, *IEEE Internet Computing*, vol.11, no.6, pp.53-59, Nov-Dec, 2007.

[6] http://java.sun.com/developer/technicalArticles/WebServices/restful/