International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 06, June 2015
Available at http://internationaljournalofresearch.org

# Building Rich Internet Applications using Google Web Toolkit

Shiv Kumar Goel[1] & Shikha Singh[2]
Asst. Prof, Deputy HOD, Master of Computer Application, VESIT, Mumbai University, India
*shivkumar.goel@ves.ac.in*
Student, Master of Computer Application, VESIT, Mumbai University, India
*shikha.singh@ves.ac.in*

## Abstract

*Traditionally, Web applications have had great limitations in the usability and interactivity of their user interfaces. To overcome these limitations, a new type of Web applications called Rich Internet Applications (RIAs) has recently appeared providing richer and more efficient graphical components similar to desktop applications. Rich Internet Applications (RIA) offer greatly enhanced usability, and allow Internet programs to rival their desktop counterparts for functionality. With the popularity of Web developmentpatterns 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. This paper presents the approach to rapid development of Web applications using GWT-based framework, as well as Ajax technologies using only Java as the programming language, which is later on compiled into pure JavaScript and deployed as regular web site. It also gives the real case applied to the development of Collateral Management System, which proves that the approach could accelerate the development process of Web-based applications.*

**Keywords-**
Rich Internet Applications; Google Web Toolkit; Rapid Application Development; AJAX; JAVA; collateral management

## 1. INTRODUCTION

Traditional Web applications developers have focused all their activity around a client-server architecture where all processing is done on the server side and a thin client which is only used to display static contents. This approach has suffered significant drawbacks and limitations, especially due to the richness of the application interfaces and the overall sophistication of the solutions that could be built and delivered.

These old-fashioned Web applications are being replaced by the so-called Rich Internet Applications (RIAs) which provide richer and more interactive user interfaces, similar to desktop applications. Moreover, RIAs provide a new client-server architecture that reduces significantly network traffic using more intelligent asynchronous requests that send only small blocks of data.

Web 2.0 is introduced as a response to growing need of interactivity which is the core assumption of contemporary user collaboration over the network. It has become online service itself. The main problem that had to be addressed with Asynchronous JavaScript and XML (AJAX) technology concerned the need for resource-wasteful whole page refreshing after every user action regardless of whether the contents have actually changed. Using this JavaScript technology, each page update with new data from server was accompanied by asynchronous request which made complex applications behave more like traditional desktop software with distributed architecture. With the increasing complexity of AJAX applications, new scalability problems arose. The biggest complication is the origin of AJAX technology. It is derived from web technologies, it was never meant to be used exclusively in large scale applications and therefore lacks tools (i.e. Integrated Development Environment (IDE), debugging etc.) that support development in the way desktop software creation is.

Google came up with new approach: why create something new instead of just taking the advantages of already known, well-founded existing technologies, methodologies and tools?

Fortunately, since the release of Google Web Toolkit (GWT) and related RIA widget frameworks such as Smart GWT, RIA development is now becoming easier and more flexible.

Google Web Toolkit is an open source Java

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 06, June 2015
Available at http://internationaljournalofresearch.org

development framework that lets developers escape the matrix of technologies that make writing AJAX applications so difficult and error prone. With GWT, programmers can develop and debug AJAX applications in the Java language using generic Java development tools of their choice. When the application is deployed to production, the GWT compiler translates user written Java application to browser-compliant JavaScript and HTML.

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, GWT introduces new structural and behavioral models in order to help developers construct Web applications that look and function like traditional desktop applications.

GWT development cycle has the following steps:

1. Using developer's favorite Java IDE to write and debug an application in the Java language, using as many (or as few) GWT libraries as it is considered useful.

2. Using GWT's Java-to-JavaScript compiler to distill application into a set of JavaScript and HTML files that can be served with any web server.

**2.** Confirming that application works in each browser that it to be supported, this usually takes no additional work.

As mentioned, GWT consists of libraries responsible for specific functionalities that may or may not be used (internationalization, communication, JavaScript Native Interface, visual interface components - widgets etc.) Most important elements are shown in Fig. 1.

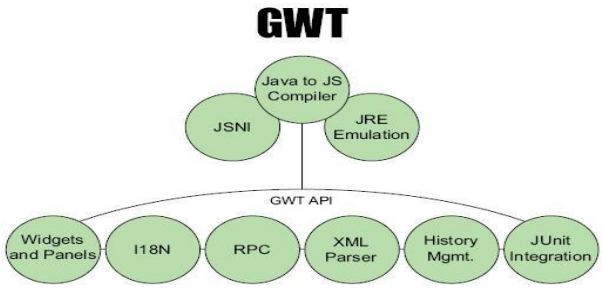


Fig. 1 - GWT component overview

## 2. APPLICATION GOALS

Existing System is an exposure & collateral management system that is used within the Stock Loan & Prime Brokerage Operations and business in Visual Studio. Collateral Management System is a client-server application, deployed on Citrix. It is not available as a standalone application on user's desktop.

The Citrix product set is part of a family of technologies termed 'server-based computing'. Server-based computing (SBC) is a technology whereby applications are deployed, managed, supported and executed on the server and not on the client. Instead only the screen information is transmitted between the server and client.

Overview of Collateral System:

- At a high level, collateral management is the function responsible for reducing credit risk in unsecured financial transactions.
- Credit risk exists in any transaction which is not executed on a strictly cash basis.
- An example of credit-risk free transaction would be the outright purchase of a stock or bond on an exchange with a clearing house.
- Examples of transactions involving credit risk include over the counter (OTC) derivative deals (swaps, swaptions, credit default swaps, CDOs) and business-to-business loans (repos, total return swaps, money market transactions, term loans, notes, etc.).
- Collateral of some sort is usually required by the counterparties in these transactions because it mitigates the risk of payment default.
- Collateral can be in the form of cash, securities (non-cash).

The proposed system aims at developing and enhancing an existing Collateral system in GWT Technology which will be a web Based application. All the users of system will be able to access the application via web.

International Journal of Research
International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 06, June 2015
Available at http://internationaljournalofresearch.org

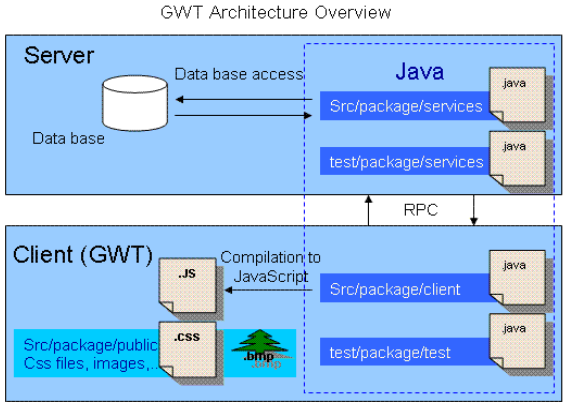


Fig. 2 – GWT architecture

Advantages of GWT.

- GWT is a java AND JavaScript. We can use jsps to create the page that our GWT module resides in. we can use spring to manage our beans and secure the client and server side of the application.
- Google web tool kit is the perfect solution for integration with old systems and re-engineering them. Application interface development (with GWT) contains a capable way to define usability for each system and subsystem, development of new GUI and re-arrangement of important components with maximum comfort for user.
- It gives users a unique opportunity to enjoy rich interface applications with a clear picture of web application interface performance and the ability to track statistics, failure elements and internal transactions of profiles.
- Because almost everything runs on the browser side we use less server processing and we can scale to more users.
- Because almost everything runs on the browser you need less server-side session information.
- No JavaScript Programming which is hard to test, and check that it works on multiple browsers.

## 3. ARCHITECTURE

Application was intended to be designed according to good practice Model-View-Controller pattern to create portable application consisting of independent elements. However, this pattern cannot be easily adapted in case of distributed AJAX application such as Collateral system as communication is asynchronous. Fig. 3 shows correlation between elements.

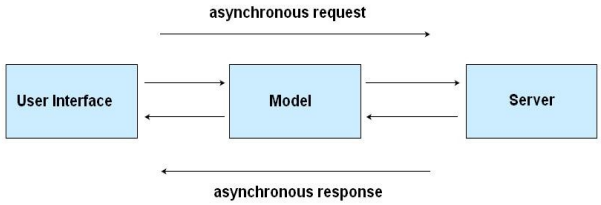


Fig. 3 Application pattern

Each request for data cannot be limited to directly operate only on nearest layer as it is only expected to receive response in some nearest future or not at all due to errors that can happen on the way in particular element (interface, network connection, business logic etc.). This creates more complex, distributed, abstract model that sits in the middle between actual components of client and server. Additionally, request must be followed by response in the form of server's callback request which then needs to be processed back in the calling method on the client side.

## 4. IMPLEMENTATION

As Fig. 4 shown, a kind of hierarchical and extensible framework is proposed in this paper. In this Fig 4, the rich client is deployed and run in the user's browser, and the layers of server-side is deployed and run in the Web application server, and the communication mode is Ajax between the two. The rich client is provided with widget by GWT based on Ajax. The framework proposed by this paper makes full use of the characteristic of GWT technology and obtains further depuration and encapsulation, which enable the framework more suitable for the design and development.

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 06, June 2015
Available at http://internationaljournalofresearch.org

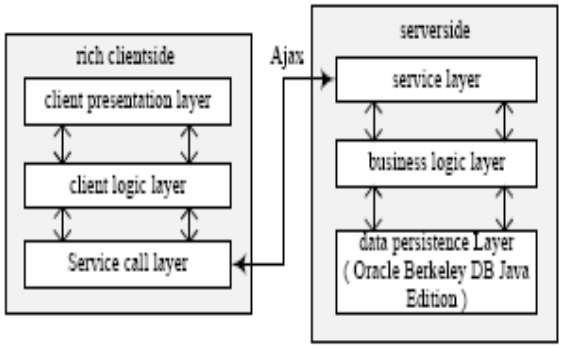


Fig. 4 System architecture

A. client presentation layer

The client presentation layer is in charge of downstage interface display of System, the implementation of which includes two parts: one is XML and second is java file using UI binder;
The basic idea is to refresh as the unit of UI Widget in the design of interface, namely the whole page is divided into some Widgets, which are separated from each other. When the data is obtained by calling background service, the web pages need to be refreshed. The smaller particle size of refreshing Widget is better, such as only refreshing a textbox. In doing so it will be useful to reduce the traffic and response time. In order to implement the above design, a global access point of UI Widget that needsto be refreshed is provided. Because GWT adopts asynchronous invoking to access service, an object whose type is Callback has been constructed in every invoking, and this object is in charge of update interface after returning the result of invoking.

B. client logic layer

In order to embody the technical features of rich client side, the functions of data cache and logic calculation should be provided in the client logic layer of client-side. Making full use of the computing capability of client-side can improve the response time of system and reduce the load of server-side. The reusability and maintainability of client-side application can be improved significantly by designing the tool class of collateral system to implement the above functions. Such as, Client Value is a class to implement data cache in client-side, whose lifecycle is a page lifecycle. Because Ajax rich client side seldom carries on page-level update, the cached data can be used for a long time. The cached data includes: login information and authority information for authority verification of client-side; the data of the previous and latter page can be saved for keeping the consistency of page switching while pagination displaying.

C. service calling layer

Service calling layer is responsible for calling service offered by the service layer of server-side and returning the result to client logic layer and client presentation layer. In other words, the service calling layer decouple between foreground logic and background logic. According to each MyService in this layer, a MyServiceCaller class is designed. Both the construction of the service object and the access of operating the service object are encapsulating, which enable client-side to call the service by MyServiceCaller class. In doing so, it is implemented to centralized control of background service access in order to maintain the code.
Taking FxRates widget as example, we explain the implementation principle of the service calling layer. The widget of client-side includes UI controls, and every control may indicate the screen event to use GWT RemoteService such as “OnClick”. Before calling RemoteService, the widget must create a DTO (a common Java EE design pattern) by its widget fields. The screen event “On Click” will use AsyncCallBack module to deal with the object returned from the class methodsgetFxRate() of server-side of the application. ForeignExchangeDTO is a POJO which is used to transmit the data from server-side to client-side. JavaScript only uses the basic data types; it must be transmitted to user's browser and remote application servers through serialization, which is a limitation that all the GWT DTO requires. All the entity class of collateral system need not implement GWT interface. They are completely isolated from the client application. To achieve this separation, a Java EE DTO design pattern is used. These DTOs contain only basic data types and are used by both packages (Services and GWT Remote Services). DTOs must implement the GWT IsSerializable interface and a GWT configuration XML file must also be created. Another important reason for using DTOs is that GWT client classes must be transformed into JavaScript code. An example is the code fragment as follows.

```
Public class ForeingExchangeDTO implements Serializable {
        final long
```

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 06, June 2015
Available at http://internationaljournalofresearch.org

```
serialVersionUID=240695217824736L;
        Date fxRateDate;
        String fxRateCurrency;
        BigDecimal fxRateValue;
        String lastUpdateUSer;
        String lastUpdateTime;
        String dataStatus;

@Override
Public String toString (){
……}

Public ForeingExchangeDTO () { }

//getter setters of variables
…… }
```

Calling service from client layer:

```
Private RPCFxRatesMaintenanceAsync
pricing=(RPCFxRatesMaintenanceAsync)
GWT.create(RPCFxRatesMaintenance.class);


pricing.getFxRates(new callback.PriceCurrCallback);

public class callback{
        AsyncCallback<List< ForeignExchangeDTO
>> PriceCurrCallback=new AsyncCallback<List<
ForeignExchangeDTO >>() {

@Override

Public void onSuccess (List<Foreign ExchangeDTO
> result){       }

Public void onFailure(Throwable caught){
}
}
}
```

D. Service layer, business logic layer.

At this point the conceptual entities have been created. The basic services methods will manage the transaction states of those entities. The next step is to create GWT Remote Services. They will invoke stored procedures to insert, update and delete data in the database. 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 a HTTP POST, call generated and managed by the GWT. The server side code for that Ajax call is found within the "server" package of the GWT application. This is code that executes within a servlet container or a Web application server and this is where we tie user actions at the browser into business logic and then the database calls to update data in the tables. The service class on the service layer must inherit from the RemoteServiceServlet class offered by GWT which is a HttpServlet. However, if the service class directly inherits RemoteServiceServlet, the global control of all the service classes will be lost. So, we must design a MyRemoteServiceServlet class which may inherit RemoteServiceServlet class, and then all the other service classes inherit from MyRemoteServiceServlet class in order to enhance the system scalability. The business logic layer focus on the system design relating to business requirement, such as the business rules setting, the implementation of the business process and etc. The business logic layer is related to the domain logic and provides support to construct service for the service layer. Taking Foreign Exchange domain logic as an example, next the interaction process between the business logic layer and database. The RPCFxRatesMaintenance interface itself does not deal with any business logic, and it is mainly used to dispose the transaction state of the special task on one or more objects of the database. If you want to add business process logic to service, the RPCFxRatesMaintImpl class must extend the RemoteServiceServlet class and implements RPCFxRatesMaintenance, and handle with all the access of the Foreign Exchange entity by providing the implementation of addFx(), updateFx(), getFxRates() and other methods. The implementation that may call the business service known as RPCFxRatesMaintImpl is in the server-side package of Collateral System, and it is just Service to implement the business logic for the specific instance of Foreign Exchange. The implementation need transform the object of ForeingExchangeDTO to the object of ForeignExchange. The database is only a set of files in the Server file system, which can be accessed by the servlet container of Web application. In the business logic layer, the application should separate the DTO object and business object through delamination and abstract. The business service need not understand the DTO class and only disposes the conceptual model object which is always in the database. The business service class is in charge of managing the conceptual model object in the data storage.

Public interface RPCFxRatesMaintenance extends

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 06, June 2015
Available at http://internationaljournalofresearch.org

```
RemoteServlet {
  Public List<ForeignExchangeDTO> getFxRates()
                throws Exception;
       public boolean addFx() throws Exception;
       ……}


Public interface RPCFxRatesMaintenanceAsync {
      public void getFxRates() throws Exception;
       public void addFx() throws Exception;
       ……}

Public class RPCFxRatesMaintImpl extends
RemoteServiceServlet implements
RPCFxRatesMaintenance {

@Override
Public List< ForeignExchangeDTO >getFxRates ()
throws Exception {
……
//Business Logic and call to database stored
procedure}

@Override
public boolean addFx() throws Exception {

……}}
```

## 5. CONCLUSION

GWT delivers all the power of true AJAX, which greatly accelerates the integration with existing Java business logic and custom data tiers. We have focused our study in the Collateral System model-driven development process that introduces models and transformations to obtain a complete RIA for the GWT framework. The presentation and user interface models are one of the most important contributions of this work. They represent the structural and the behavioral aspects of the RIA user interface, allowing to define both simple and multi-page applications. GWT developed application increases the flexibility of the system, enhances platform scalability, making local data changes not affect the entire platform.
Since the update of GWT framework is fast and new widgets emerge continually, Collateral System on GWT should keep updated to keep up with GWT and improving the look and feel of the application.

## 6. 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]http://www.bobsguide.com/guide/collateral-management-systems.html

[6] www.**gwt**project.org/

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

[8]Bo Song, Jie Liu, and Chuan-Sheng Zhou, "Implementation of J2EEData Persistence Tier with TopLink", Microelectronics & Computer, Vol.23, No.8, pp.132-135, August, 2006(in Chinese).

[9]Bo Song, Jing Zhao, "Research on Network Teaching System Basedon Open Source Framework", IEEE Ninth International Conferenceon Hybrid Intelligent Systems, Vol.1, pp.28-32, August, 2009.

[10] Wei Fang, Yong Sun, and Zhi-Ming Cui "Research and Applicationof J2EE's Data Persistence Layer", Computer Technology andDevelopment, Vol.17, No2, PP.68-91, February, 2007(in Chinese).

[11] P. Chaganti. "Google Web Toolkit: GWT Java Ajax Programming", American: Packt Publishing, pp.21-24, 2007(in Chinese).