J. Software Engineering & Applications, 2010, 3: 73-80
doi:10.4236/jsea.2010.31009 Published Online January 2010 (http://www.SciRP.org/journal/jsea)

Integrated Web Architecture Based on Web3D, Flex and SSH

Wenjun ZHANG[1,2]

[1]Department of Computer Science & Technology, China Women University, Beijing, China; [2]Research Institute of Applied Computer Technology, China Women University, Beijing, China.
Email: voicefromzhwj@yahoo.com.cn

Received September 9th, 2009; revised October 9th, 2009; accepted October 19th, 2009.

ABSTRACT

Focusing on the problems occurred in traditional 2D image-word-based web applications, the author put forward concept of integrating Web3D, Flex and SSH technologies to create advanced "3D Virtual Reality & RIA" web application architecture, researched mechanisms of their architectures, and implemented their integration and communication & interaction: Flex and Struts2 via XML, Flex and Spring & Hibernate via BlazeDS, Flex and Web3D via JavaScript. The practice has shown that the integrated web architecture based on Web3D, Flex and SSH is effective and valuable.

Keywords: *Web Architecture, Web3D, Flex, SSH, RIA, BlazeDS*

1. Introduction

E-commerce, e-government and enterprise-informationization (CGE) are web applications in relevant domains, and have been quickly developed with the improvement of web technology. The transaction mode between customers and suppliers has been changed from direct purchase in stores to shopping in internet; and information management system within enterprises has become integrated on internet covering SCM, CRM and ERP.

However, CGE encounters some serious problems as it develops. For example, products in e-commerce websites can only be exhibited by images and words. This kind of manifestation can't express completely structures and functions of the products, which reduces the client's desire to purchase. Tedious interactive form, poor efficiency and unsatisfactory user experience widely exist in current CGE applications.

The problems are essentially due to the weakness of the traditional web technology based on HTML web pages—thin client B/S (Browser/Server) mode. HTML web pages only display content, but not contain the script. The client-side can only requires data through request and response session because all datum host on the server-side. The content from server contains not only data but also a lot of redundant display formats. Unlike desktop applications, browsers (IE, Firefox) are not equipped with multifunction controls such as DataGrid, Tree, and PieChart. The codes in presentation, transaction logic and data persistence layers, are tightly coupled together which result in low reuse, high couple and difficult maintainability.

Because the traditional web technology seriously bottlenecked progression in CGE applications, new technologies such as Web3D, RIA (Rich Internet Application) and SSH (Struts, Spring, Hibernate) are introduced into web application development. In other words, Web3D is applied to simulate product shape and functions in 3D and interact with customers on the client-side; Flex technology, a kind of RIA, realizes business process, human-computer interaction and data visualization (charts, curves) on the client-side; SSH, three popular web design frameworks, implements transaction logic and data persistence on the server-side which develops low-coupling codes. The web architecture studied in this article features 3D virtual reality, high interaction, abundant user visual experience, and low coupling & high maintainability.

In the article, the features and application development methods on Web3D, Flex and SSH technologies in CGE applications are studied. The advanced and integrated web architecture (shown as Figure 1) based on Web3D, Flex and SSH is constructed in order to convert the traditional thin client B/S mode featured by Image-Word into advanced rich client mode featured by 3D Virtual Reality-RIA.

2. Web3D Technology and Application in E-Commerce & Data Visualization

Web3D technology enables e-commerce with three

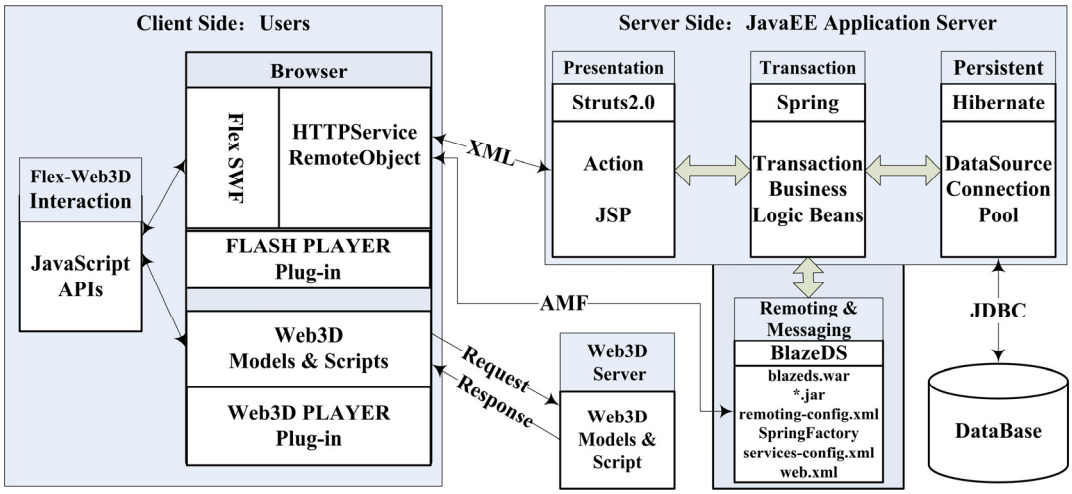

Figure 1. Integrated web architecture based on Web3D, Flex and SSH

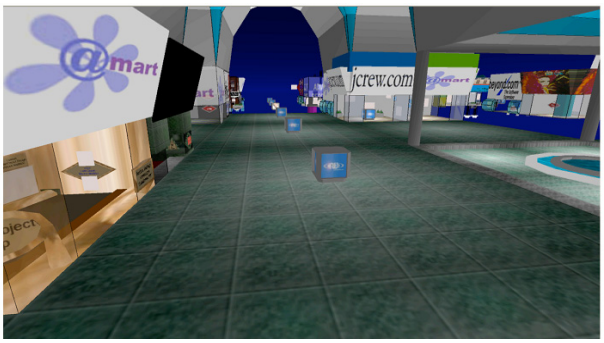

Figure 2. RIA & Web3D virtual shopping city

dimensional presence of the products and the scenes. Real product features such as shape, functions and scenes are simulated: customer can navigate in virtual 3D-store [1–2] shown in Figure 2, interact with 3D virtual products, and purchase goods. Their virtual shopping experiences are almost as vivid as in real stores. The success rate in sales is expected to be much higher than before because customer can feel the goods in this 3D store, and it becomes unnecessary for them to check products in real stores before making purchase decisions. In the following sections, the core technologies for Web3D and application development will be presented.

2.1 Web3D Implementation Technology and Solution

According to implementation methods of virtual 3D models and scenes the technologies can be classified into two categories [3]: Model-based Web3D and Image-based Web3D.

1) Model-based Web3D. It is also called landscape-based geometric. It is to construct virtual models and scenes with geometric entity. The geometric entity model is constructed by different 3D authoring software according to computer graphics, and is rendered real time. Human-computer interaction is implemented by adding event response. Model-based Web3D mainly includes X3D, Cult3D, Viewpoint and O3D released newly by Google.

2) Image-based Web3D. Its core technology is based on panorama, which is a kind of closed-view shot around from some point in space. Image-based Web3D is also classified into cylindrical panorama and spherical panorama. The panorama may be required through photograph and computer-3D rendering. Image-based Web3D mainly includes Java3D and Flash3D.

2.2 Web3D Application Framework

Web3D player engine or plug-in is previously installed in browser in the client PC before 3D models are rendered on the client-side. Web3D application framework is shown in Figure 1. Web3D models & scenes and built-in scripts are downloaded from Web3D application server, interpreted, rendered, and interacted with users by Web3D player engine [4–5].

2.3 Web3D Design Method Based on Separation between Model and Function Script

During authoring Web3D models one design method is often adopted based on separation [1] between model and function script. This approach is to author Web3D virtual product models and scenes by ViewPoint, Cult3D or O3D, and use XML, Java or other script languages to program scripts for the authored 3D models so as to change 3D model view with user's actions and to response different events such as click, mouse movement and drag-drop generated from users. Thus, the 3D models are independent from their scripts.

The programming method of separation between model and function script can improve flexibility, reus-

ability and scalability of Web3D product models.

3. Flex Technology and Application in Web & Data Visualization

Most of the current CGE information systems are B/S thin client application mode based on HTML pages. With increasing complexity this application mode is not longer able to meet the requirements of providing interactive and rich user experience.

Now RIA, the next generation web technology [6], has been developed, which combines interactive user experience with the deployment flexibility. The rich client technology in RIA can connect with existing back-end applications by asynchronous communication between client and server. It is a service-oriented model with good adaptability. It improves the user interaction of web application, and offers abundant user experiences.

Flex [7] is one kind of RIA technology, a framework for creating RIA based on Flash Player. Its core is MXML, a markup language based on Extensible Markup Language (XML) that makes it really easy and efficient to create applications.

Flex offers highly visual, fluid, and rich experience and user interface components. When JEE and Flex are integrated together in a Web application, the best of both worlds are combined. JEE provides the power and stability on the server-side, while Flex and Adobe Flash Player make the rich, dynamic user interfaces possible on the client-side.

3.1 Flex Framework Technology and Development Process

The Flex framework [8,9] shown in Figure 3 is synonymous with the Flex class library and is a collection of ActionScript classes used by Flex applications. The Flex framework is written entirely in ActionScript classes, and it defines controls, containers, and managers in order tosimplify the process of building RIA. The four main parts in the Flex framework are represented in the following.

1) MXML. MXML is an XML-based markup language that primarily describes screen layout. In this respect it is similar to HTML. Using MXML tags, you can add components such as form controls and media playback to layout containers such as panel. In addition to screen layout, MXML could be used to describe effects, transitions, data models, and data binding. MXML is so robust that it is possible to build many applications entirely with MXML. Flex Builder enables developers to construct MXML with a What-You-See-Is-What-You-Get approach--build basic Flex applications without writing any code.

2) ActionScript. ActionScript is the programming language understood by Flash Player and is the fundamental engine of all Flex applications. Even though MXML simplifies screen layout and basic tasks, ActionScript can not only do everything that MXML can do, it can also do many things that MXML cannot do. For example, ActionScript can respond to events such as mouse clicks, while MXML can not. Although it is possible to build an application entirely with MXML or entirely with ActionScript, it is more effective to build applications with both MXML and ActionScript and the two work well together. MXML is best suited for screen layout and basic data features, while ActionScript is best suited for user interaction, complex data functionality, and any custom functionality not included in the Flex class library. ActionScript is supported natively by Flash Player, and does not require any additional libraries to run--all native ActionScript classes are packaged in the Fash package or in the top-level package. In contrast, the Flex framework is written in ActionScript, but those classes are included in a .swf file at compile time.

3) Flex Class Library. Flex framework defines the Flex class library. It consists of predefined components such as controls, containers, data components, and Flex Data Services for communication with application back-end server.

4) Flex Data Services. They provide the remoting and messaging foundation for connecting a Flex-based front-end to JEE back-end services, and transport data between the client and server. BlazeDS, a kind of Flex Data Services technology, will be represented in the following

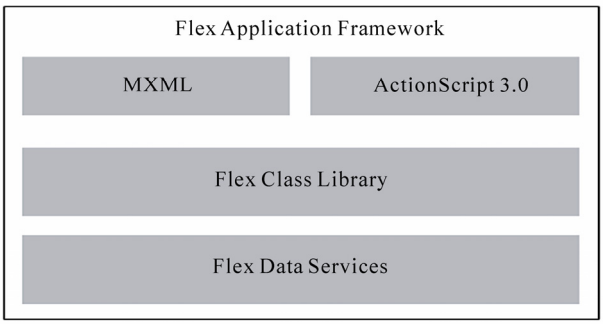

Figure 3. Flex framework

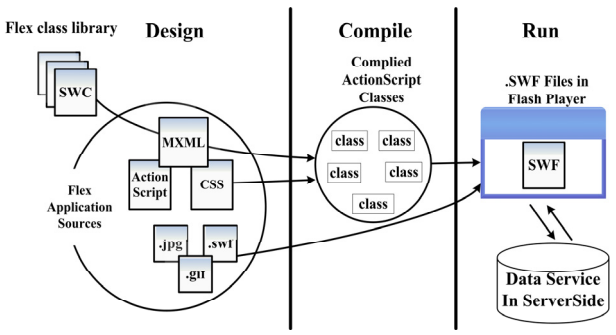

Figure 4. Development process of flex applications

section.

The development process [10] of Flex applications is shown in Figure 4.

3.2 Strategies of Flex Access to Server-Side Applications

Four strategies [10] to access server-side applications:

1) HTTPService. The HTTPService component sends HTTP requests to a server, and consumes the response. Although HTTPService is typically used to consume XML, it can also be used to consume other types of responses. HTTPService is similar to the XMLHttpRequest component available in Ajax.

2) WebService. The WebService component invokes SOAP-based web services. It's similar to HTTPService.

3) RemoteObject. The RemoteObject component directly invokes methods of Java objects deployed in application server, and consume the return value. The return value can be a value of a primitive data type, an object, a collection of objects, etc. In distributed computing terminology, this approach is generally referred to as "remoting". This is also the terminology used in Spring to describe how different clients can access Spring beans remotely.

4) BlazeDS. In addition to the RPC-type services described above, BlazeDS , a kind of the Flex Data Management Services, provides an innovative and virtually code-free approach to synchronize data between the client application and the middle-tier.

In the following section, the article will discuss in more details about Remoting and BlazeDS because they enable the tightest integration with Spring and Hibernate.

4. SSH Technology and Application on the Server-Side

For years, JEE has been used to develop server-side web applications. Normally these applications are developed by Java Server Pages (JSP) and Servlet, which dynamically insert server-side data into HTML for the user interface to create dynamic data-driven applications. This development technology results in highly tight-coupling codes, which make reuse and maintainability very low. Therefore nowadays, SSH (Struts, Spring, Hibernate), three popular web design frameworks, is widely applied to develop web application on the server-side in order to program looser-coupling codes.

Struts framework, based on MVC-2 architecture, is an open-source framework for developing the web applications in JEE, which extends Java Servlet API. Struts is a robust architecture and can be used for the applications of any size. Struts is often applied to develop and implement presentation layer of web applications with a set of cooperating classes, servlets and JSP tags that make up a reusable MVC-2 design.

Spring [11] is one of the most popular Java frameworks. The foundation of the Spring framework is a lightweight component container that implements the Inversion of Control (IoC) pattern. Using an IoC container, components do not instantiate or even look up their dependencies (the objects they work with). The container is responsible for injecting those dependencies when it creates the components (hence the term "Dependency Injection" also used to describe this pattern). This will result in loosing coupling between components. The Spring IoC container has proven to be a solid foundation for building robust enterprise applications. The components managed by the Spring IoC container are called Spring beans. Spring is often applied to develop and implement transaction logic layer of web applications.

Hibernate is a pure Java object-relational mapping (ORM) and persistence framework that allows you to map plain old Java objects to relational database tables using (XML) configuration files. Its purpose is to relieve the developer from a significant amount of relational data persistence-related programming tasks. Hibernate is mainly applied to develop and implement persistence layer of web applications. The main advantage of ORM like Hibernate is that it shields developers from messy SQL. Apart from this, ORM provides following benefits: improved productivity, performance, maintainability and portability.

5. Integration & Interaction among Web3D, Flex and SSH

In the following section the configurations and communication codes among Web3D, Flex and SSH are implemented for their integration-Flex and Struts2 via XML, Flex and Spring & Hibernate via BlazeDS, Flex and Web3D via JavaScript.

5.1 Integration & Interaction between Flex and Struts2

Apache Struts is an incredibly popular open source framework for building Servlet/JSP based web applications on the MVC design paradigm. The view layer in Struts is HTML, but the isolation between the view and rest of the pieces ensures that Flex can easily replace HTML. The migration or creation of such an application with a Flex front-end requires transforming the existing view layer to XML-formatted output.

The communication process between Flex and Struts 2.0 [12] is like this: on the Flex client-side the HttpService or WebService component sends URL request such as http://localhost:8080/flexstruts2/login.action?name=zhangwenjun&password=iloveyou. Then Struts2.0 framework on the server-side receives and dispatches the request to interceptors and actions, and returns values to the result tags in struts.xml configuration file to switch the different JSP files. In fact the JSP files in structure have be-

come dynamic XML files. They are compiled, executed and downloaded by application server such as Tomcat for the HttpService or WebService component to receive and decode XML data to refresh Flex user interfaces. The following codes are XML-formatted JSP file. The complete integration architecture between Flex and Struts2 is shown in Figure1. The configurations of Struts 2.0 remain the same.

```
<?xml version="1.0" encoding="UTF-8"?>
<% ..... %>
<persons> <% while ( rs.nexe() ){ %>
  <person>
    <name><%=rs.name %></name>
    <age><%=rs.age %></age>
  </person><% } %>
</persons>
```

5.2 Integration & Interaction between Flex and Spring

Spring BlazeDS Integration [13] is a top-level solution for building Spring-powered Rich Internet Applications using Adobe Flex for the client-side technology.

BlazeDS is an open source project from Adobe that provides remoting and messaging foundation for connecting a Flex-based front-end to JEE back-end services. BlazeDS contains configurable channels that transport data between the client and server. Though it has previously been possible to use BlazeDS to connect to Spring-managed services, it has not been in a way that feels "natural" to a Spring developer, requiring the extra burden of having to maintain a separate BlazeDS xml configuration. Spring BlazeDS Integration turns the tables by making the BlazeDS MessageBroker a Spring- managed object, opening up the pathways to a more extensive integration that follows "the Spring way".

BlazeDS clients use a message-based framework provided by BlazeDS to interact with the server. On the client side of the message-based framework are channels that encapsulate the connection behavior between Flex client and BlazeDS server. Channels are grouped together into channel sets responsible for channel hunting and channel failover. The following illustration in Figure 5 shows the BlazeDS architecture.

The BlazeDS server is contained in a JEE web application. A Flex client makes a request over a channel routed to an endpoint on the BlazeDS server. From the endpoint, the request is routed through a chain of Java objects that includes MessageBroker object, a service object, a destination object, and an adapter object. The adapter fulfills the request either locally, or by contacting a back-end system or a remote server such as Java Message Service (JMS) server. The following illustration also shows the BlazeDS server architecture.

The idea behind Spring IoC is to let the container instantiate components (and inject their dependencies). By default, however, components accessed remotely by a Flex client are instantiated by Flex destinations on the server-side. The key to the Flex/Spring integration is therefore to configure the Flex destinations to let the Spring container take care of instantiating Spring beans. The Flex Data Services support the concept of factory to enable this type of custom component instantiation. The role of a factory is simply to provide ready-to-use instances of components to a Flex destination (instead of letting the Flex destination instantiate these components itself). The supporting files available include a factory class (Spring Factory) that provides Flex destinations with fully initialized (dependency-injected) instances of Spring beans.

How does Flex access Spring beans? If Flex clients can remotely access Java objects, and if Spring beans are Java objects, we still need following steps of configurations to access Spring beans from Flex clients: 1) To register the Spring Beans in applicationContext.xml; 2) To configure the Flex Remoting Destination in applicationContext.xml; 3) To register the Spring Factory in services-config.xml. Notice that we provide the name of the Spring bean as defined in applicationContext.xml as the source.

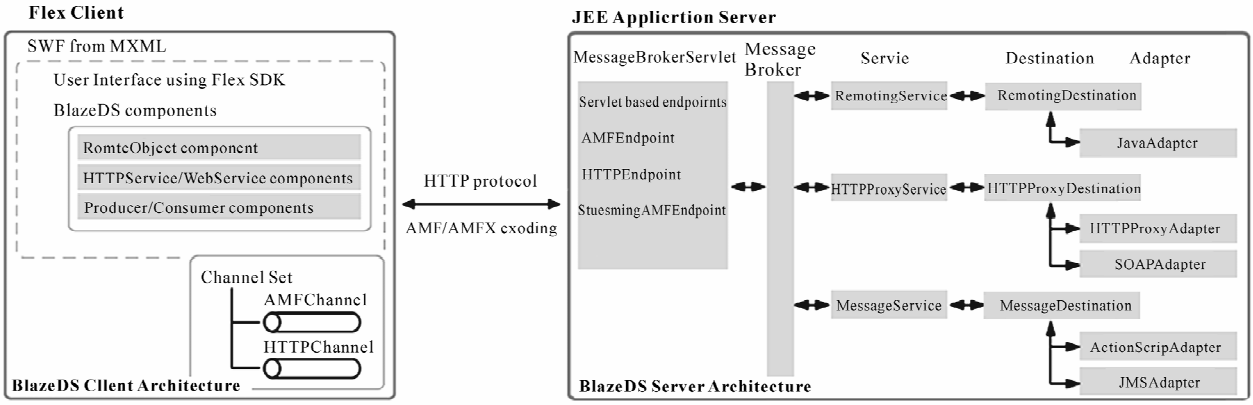

Figure 5. Blaze DS client & server architecture

5.3 Integration & Interaction between Flex and Hibernate

If Hibernate is used for web application it will be better implementing our own DataAccess objects and providing public methods of these objects to our clients by a Web-service. Hibernate persists Java objects.

Hibernate and Flex front-ends can be easily connected by using BlazeDS DataManagement Service. In Java developed classes can be accessed in the Flex application using an ActionScript class with the same setter- and getter-methods. This ActionScript class represents the Java class on the client-side. Any changes you make at the ActionScript class will be directly affecting the Java class on the server. There is no need to develop DataAccess classes. All create-, update-, and delete-operations are handled by DataManagement Service. Behind the scenes the class flex.data.adapters.JavaAdapter is taking care of the CRUD-Operations.

The hibernate.cfg.xml conatains common settings for the database connection, the SQL dialect and a few more which are not explained in detail here. In the myclass.hbm.xml we map the class myclass. The mapping file has a root tag called <hibernate-mapping>. Within this tag the classes are mapped for the SQL database. In more complex applications every class should have its own mapping file.

Configuration of the RMTP-Channel and the Destinations in Figure 5: Like all configuration files for the BlazeDS we will have the DataManagement configuration files in the folder [CONTEXT-ROOT]/WEB-INF/flex of web application server, in this case the Tomcat. The important files for the application are services-config.xml and data-management-config.xml. In the services-config.xml we have to declare the RTMP Channel and the file contains this entry by default.

In the data-management-config.xml we need to declare the Destinations for our developed Java classes. In the <destination> tag we give this destination an ID which will be used later by the Flex application to refer to this particular destination.

To communicate with the Java objects at the Tomcat we need to implement ActionScript classes which are the counterpart to the Java classes—they have the same properties as the Java classes. We declare the properties public so there is no need to write getter- and setter-methods. By definition the ActionScript classes must contain an empty constructor. The classes are declared as managed and we set an alias to the corresponding Java class.

In the <mx:DataService> tag in DataService component in the Flex application, we set the property "destination" to the name of the destination as we declared it in the data-management-config.xml. The DataService component also provides methods like deleteItem() and createItem() of the typical methods of Data Access Objects. Note that we never declared or implemented these methods. This work is already done by the Flex DataManagement Service.

5.4 Communication & Interaction between Flex and Web3D via JavaScript

In application interaction and communication between Flex and Web3D models on the client-side are required. Flex sends data to Web3D, and the Web3D models and scenes change with the received data. On the other hand, Web3D can also send data to Flex, and the visual chartings or tables or other components in Flex change with the received data. In addition, the Flex can forward the received data to the server-side.

How can we implement communication & interaction between Flex and Web3D? It is known that Flex runs in Flash Player, and that Web3D models and scenes run in Web3D Player. Both Flex and Web3D are embedded in HTML web pages in browser, and the direct communication between Flash Player and Web3D Player is impossible. However both Flex and Web3D can communicate with JavaScript in web browser, therefore JavaScript can be served as an intermediary between Flex and Web3D. That is, Flex communicates and sends data to JavaScript, then it receives and forwards the data to Web3D; on the other hand, Web3D can send data to Flex via JavaScript shown in Figure 1.

5.4.1 Calling JavaScript Functions from Flex

The External Interface API is used to call JavaScript functions from Flex and build wrappers to call from Flex.

1) Flex Codes. In Flex program shown in the following example an ActionScript function is added into the Script tags, and "flash.external.*" package needs to be imported. If ExternalInterface is available, the ExternalInterface.call will be executed. Then the information will be sent to a JavaScript function called "displayPerson", whose argument is the selectedItem in a DataGrid. The status label saying "data sent" will be updated. If the ExternalInterface is not available an error message will be displayed in the label. The scripts look like this:

```
public function jsDisplayPerson():void{
  if (ExternalInterface.available) {
    ExternalInterface.call("displayPerson", dgPeople.
selectedItem);
    lblMessage.text = "Data Sent!";}
  else{lblMessage.text = "Error sending data!"; }}
```

2) JavaScript Codes. In HTML page the JavaScript code receives and displays the "person" sent from Flex. In a set of JavaScript tags the displayPerson function is created: note that the name has to match perfectly the ExternalInterface.call function in Flex. Firstly the JavaScript function checks whether it gets a null object, and if it is null an alert will be displayed. Then we just use the

object passed as a JavaScript object and reference the appropriate DataGrid columns using JavaScript object property syntax. Finally the JavaScript forwards the object to Web3D. In HTML script tags the codes can be written as below.

```
function displayPerson(person){
  if(person = = null){
    alert("Please select a person, or maybe I screwed up.");}
  else{
document.getElementById('nameDisplay').innerHTML = person.Name;
document.getElementById('ageDisplay').innerHTML = person.Age;
document.getElementById('sexDisplay').innerHTML = person.Sex;}}
```

5.4.2 Calling Flex Functions from JavaScript

1) Flex Codes. To call Flex functions from JavaScript via ExternalInterface, the first step is to add some codes to the application startup to initialize Flex functions so that it is accessible through external calls.

```
<mx:Application xmlns:mx=http://www.adobe.com/2006/mxml initialize="initApp()">
```

The initApp ActionScript function checks if the ExternalInterface is available and adds a callback for an ActionScript function. This function is externally referred to as "addPerson" and it maps the internal function called addPerson. The initApp function is added in ActionScript Script tags and shown below:

```
public function initApp():void{
  if (ExternalInterface.available) ExternalInterface.addCallback("addPerson", addPerson);}
```

Now the only thing left is to create the function "addPerson" in Flex, which adds persons to the DataGrid. This function takes three arguments: name, age, and sex.

```
public function addPerson(name:String, age:String, sex:String):void{
  (dgPeople.dataProvider as ArrayCollection).addItem({Name: name, Age: age, Sex: sex});}
```

2) JavaScript Codes. After the MXML and AactionScript completed the JavaScript function grabs values from Web3D and calls the Flex function using those values as arguments. Then function getFlexApp ('Flex JSApp') is used to call the Flex function from JavaScript. The JavaScript tag is the following:

```
function addPerson(){
var name = document.getElementById('txtName').value;
var age = document.getElementById('txtAge').value;
var sex = document.getElementById('selSex').value;
getFlexApp('FlexJSApp').addPerson(name, age, sex);}
```

This getFlexApp function in the JavaScript tag actually returns the Flex application embedded in the web page and takes into account various types of browsers. This function returns the appropriate reference, depending on the browser.

```
function getFlexApp(appName){if (navigator.appName.indexOf ("Microsoft") !=-1){return window[appName];} else {return document[appName];}}
```

5.4.3 Communication between Web3D and JavaScript

There are many polular Web3D technologies such as Unity3D, Flash 3D, and Google released O3D technology not long ago. In the following section we will discuss O3D technology and the communication between O3D and JavaScript.

O3D [14] is an open-source JavaScript API for creating interactive 3D graphics applications. O3D extends application JavaScript code with an API for 3D graphics. It uses standard JavaScript event processing and callback methods. An O3D application is contained in an HTML document. The main code for the O3D JavaScript application is contained in a <script> element inside the <head> element of the HTML document. Typically, when the HTML page is finished loading, the O3D init() function is called and executed automatically.

Because O3D is implemented by JavaScript and runs in HTML browser, it is very easy for O3D to communicate with Flex via JavaScript. The O3D JavaScript code of communication with Flex is very similar to the above code between Flex and JavaScript.

6. Conclusions

Several popular web frameworks, namely Web3D, Flex and Struts2-Spring-Hibernate, were studied for this interoperability research. The author researched how these frameworks can work together well, and created their integration architecture. Web3D was applied to simulate 3D shape and functions of products and interact with customers on the client-side; Flex was used to implement business process, rich user interfaces and data visualization on the client-side; SSH, three popular web design frameworks on the server-side, was adopted to realize transaction logic and data persistence so as to develop low-coupling codes. More importantly, the author programmed their codes of integration and communication & interaction: Flex and Struts2 via XML, Flex and Spring & Hibernate via BlazeDS, Flex and Web3D via JavaScript. All research findings were applied into an application demo named "RIA & Web3D Virtual Shopping City". The practice has shown that the architecture based on Web3D, Flex and SSH is effective and valuable.

7. Acknowledgments

This research project has been sponsored by "Key Research Fund Project in 2009" under China Women University.

REFERENCES

[1] W. J. Zhang, "Research of Web3D technology application in e-commerce," in the 5th China Conference on Software Engineering, Beijing, China, Vol. 44, pp. 225–227, November 2008.

[2] M. Zhang, Z. H. Lu, and X. L. Zhang, "Research and application of the 3D virtual community based on WEBVR and RIA," Computer and Information Science, Vol. 2, No. 1, pp. 8–15, February 2009.

[3] F. Zhang and W. W. Wang, "The analyzing about Web3D virtual reality technology," Friend of Science Amateurs, Vol. 5, pp. 130–131, May 2008.

[4] S. Chen, "Interaction design of Web3D based VR on internet," Packaging Engineering, Vol. 29, No. 4, pp. 84–86, April 2008.

[5] D. Brutzman and L. Daly, "X3D: Extensible 3D graphics for web authors," Elsevier Inc, 2007.

[6] J. Lott and D. Patterson, "Advanced actionscript 3 with design patterns," Peachpit Press, 2006.

[7] Adobe Systems Incorporated, Flex 3 help, 2009. http://livedocs.adobe.com/flex/3/html/help.html?content =profiler 3.html.

[8] Adobe Systems Incorporated, Flex 3 language reference, 2009. http://livedocs.adobe.com/flex/3/langref/.

[9] Adobe Systems Incorporated, Flex 3 data visualization developer guide, 2008. http://livedocs.adobe.com/flex/3/datavis_flex3.pdf.

[10] Adobe Systems Incorporated, Flex 3 developer's guide, 2009. http://livedocs.adobe.com/flex/3/html/help.html? content=Part2_DevApps_1.html.

[11] Walls. C, "Spring in action," 2nd Edition, Greenwitch Manning, 2008.

[12] W. J. Zhang, "Research of RIA design pattern based on Flex, Spring and Hibernate," in the 5th China Conference on Software Engineering, Beijing, China, Vol. 44, pp. 126–128, November 2008.

[13] J. Grelle, "Spring BlazeDS integration reference guide," March, 2009. http://static.springframework.org/spring-flex/docs/1.0.x/reference/html/index.html.

[14] Google, O3D developer's guide, 2009. http://code.google.com/intl/zh-CN/apis/o3d/docs/devguideintro.html.

