One way of using Ajax components to realize asynchronous communication in web service

IlNam Ri[1,*], SongIl Choe[2], Hun Kim[1]

1 College of Information Science, **Kim Il Sung** University, Pyongyang, Democratic People's Republic of Korea

2 Department of Information Science, HuiChon Industry University, HuiChon, Democratic People's Republic of Korea

Corresponding author. * E-mail address: ring15822944219@163.com

Abstract- Ajax (Asynchronos Javascript And XML), one of the world's most widely used Web 2.0 technologies, is devoid of the traditional Web page approach.(1) This technology is becoming an indispensable element in Web apps as it supports asynchronous communication that allows the user to proceed with the conversation.(3,4)

Today 's reality is getting closer to the virtual reality by the dissemination of intelligent devices, the new Internet of Things, cloud computing and the development of information society.

In the field of business services, we also need to improve the size and service quality of our web apps, and Ajax technology is constantly expanding.(5)

In the fields of stocks, finance, auctions, etc. that deal with large-scale, real-time data, it is important for business service providers to transmit information to users as soon as possible.

This Paper describes one way of using Ajax components to realize asynchronous communication of Web service providers on the Internet.

Ajax components consist of an Ajax core that supports Epoll and an Ajax library that supports asynchronous communication, and provides an application interface to define and implement various push functions.(2)

Web service applications developed using Ajax components can provide various push services by using asynchronous communication with client by using Epoll method.

.**Keywords-** Ajax, Ajax component, Web service, Server Push

1. Introduction

Since the introduction of Web 2.0, Web application development has evolved from a static Web site into a tInteractive Web service program based on a interaction. (3)

One of the most popular technologies for creating interactive web service applications is Ajax (Asynchronous Javascript and XML), which appeared in 2005 (Garrett 2005) and now runs a number of Web sites supporting Ajax technology.(1)

Ajax allows asynchronous requests after the page is loaded so that the JavaScript code can update some of the pages in the browser, effectively updating the data without having to reopen the entire page..

Today, as the demand for fast and accurate information such as the economy, construction, commerce, technology dissemination, and new news is increasing, the environment where information is actively provided on time is important without the request of the user..

Polling in the previous period is a representative technology for realizing the service push in the client / service model. It is a method that allows the client to send the request implicitly to the service provider so that the information of the service provider is obtained in a timely manner when the service information is modified .

Polling has been widely used because it is easy to implement in general HTTP based applications.It is not efficient due to frequent requests to exchange network with continuous liveness of service resources..

However, with the introduction of Ajax technology, Push technology such as Long Polling (6) and HTTP Streaming (7)are being used to improve the problem of polling and to realize effective service push.

Typically, there are Pushlet, Adobe's BlazeDS (9), and Weelya's APE (Ajax Push Engine).

Although these frameworks provide various functions and high performance, they do not provide a convenient environment for the developers or they do not support the development of a wide range of Web service applications due to the dependency on the execution environment.

In the paper to solve these shortcomings, we will discuss how to use Ajax components to support the development of push services for Web services.

The Ajax component consists of an Ajax core for developing a Push service that supports Epoll and an Ajax library for a push request to perform Ajax-based asynchronous communication.

The Ajax core part uses the non-blocking I / O (JavaNIO) to support Epoll, and at the same time it runs independently without being dependent on the execution environment.(8)

It also supports the infrastructure for convenient management of channels and user information for pushing information.

The Ajax library is bundled with the Push service and provides a variety of application interfaces to take advantage of Push service.

Ajax component based web service application programs can efficiently manage many referrals based on Epoll, and can provide push service with Ajax - based asynchronous communication.

The composition of the paper follows the introduction, and in Section 2 deals with the techniques and use of the previous period to realize the service push.

Section 3 describes the design and development of the Ajax core and Ajax library, and deals with protocols for effective communication.

Section 4 describes the conclusion.

2. Analysis of Prior Research

2.1 Server push technology

Server push is a communication technology in which the service provider actively provides information without the exact request of the client.

Table 1 shows representative methods for realizing server push technology in the Internet.

method	Characteristic
Long Polling	· Used in asynchronous frameworks such as Ajax. · When the server receives a request from the client, it maintains the association, responds with an event that the information changes, and ends the HTTP Tranjaction. · When the client receives the response, it sends the request again.
HTTP Striming	· If the information change event occurs while maintaining the connection between the server and the client like real-time Striming, the data is transmitted using the HTTP Chunked method. · Realization and exception handling are difficult and incompatible.

Table 1. Major Server Push Techniques

2.2 Framework for server push

As the necessity of pushing the server has been emphasized, a framework has been developed to effectively develop server push-based applications.(2,6)

Pushlets is a Sevlet-based Push framework that can provide information from a Java object on the server to a client's DHTML (DynamicHTML) without a PlugIn such as a Java Applet.

Adobe BlazeDS is an execution environment for servers that communicate and provide information to framework-based clients such as Flex or AIR

Pushlets and BlazeDS support asynchronous communication between server and client based on HTTP streaming.(7)

APE is an open-source push engine that provides real-time push service based on Ajax and consists of APE server and APE client.

The APE server operates as an Epoll-type Comet server and provides effective information asynchronously with APE clients.

Epoll handles multiple clients at high speed by providing Edge-trigger and Level-trigger face-to-face processing methods based on event notification.Table 2 shows the main features of the APE.

APE server	APE client
·Write in C language ·Based on distribution / subscription model · Linux, BSD & Mac OS support · E-poll Driven Lyon · Complete asynchronous guarantee Server-side JavaScript support · Batch-in message quering system · Support scalable server realization	· Based on Mootools · Client can use Socket API · Various browser support · Framework support for PlugIn · Session Management & Multi Channel Support · Transfer method support

Table 2. Major Features of APE

3. Develop Ajax components for Web business services

3.1 Configuration Ajax components

The Ajax component is a framework for providing push service by using the long polling method and consists of Ajax core for business server and Ajax library for client development.

Developers can use the Ajax core and Ajax libraries to realize the server and client of the push application, respectively, and the general user can easily use the push service of the server through this client.

The operation process of the Push App based on the Ajax component is as follows.

(1) Request information from the user
(2) Ajax library works and requests to push server
(3) Push server's request verification and client reconnection information storage
(4) Server information change event occurred
(5) Push information to connected clients
(6) Processing Ajax libraries after receiving information from the client
(7) Repeat steps 2 through 7 until the user cancels the request.

When requesting the information requested by the user, the Ajax library uses the Ajax XHR

(XMLHttpRequest) (9) object to connect to the push server based on the Ajax core and to request information.

The XHR object provides a long polling method that enables the client to communicate with the server asynchronously while operating independently of the user interface in the client. The XHR object uses Java Script Object Notation (JSON)(8,10) object for information representation and exchanges information with the server.

The Ajax library defines the push service by applying XHR object and defines various functions to realize it, and provides the client development environment of the push program.

The Ajax core handles client connections and requests with the Java NIO Selector class. It also uses a communication channel called Channel as a means of classifying and storing connected clients.

The channel is defined by the server's XML metafile and can be dynamically created and modified by the user. Connected clients are represented by objects and stored on the server's specified Channel.

When the information to be provided to the corresponding channel occurs, the connection object stored in the channel is used to transmit information to the client.

The Ajax core provides push server development by providing functions to define these channels and to provide information effectively.

Figure 1 shows the overall structure of the Business Service Push application based on Ajax components.

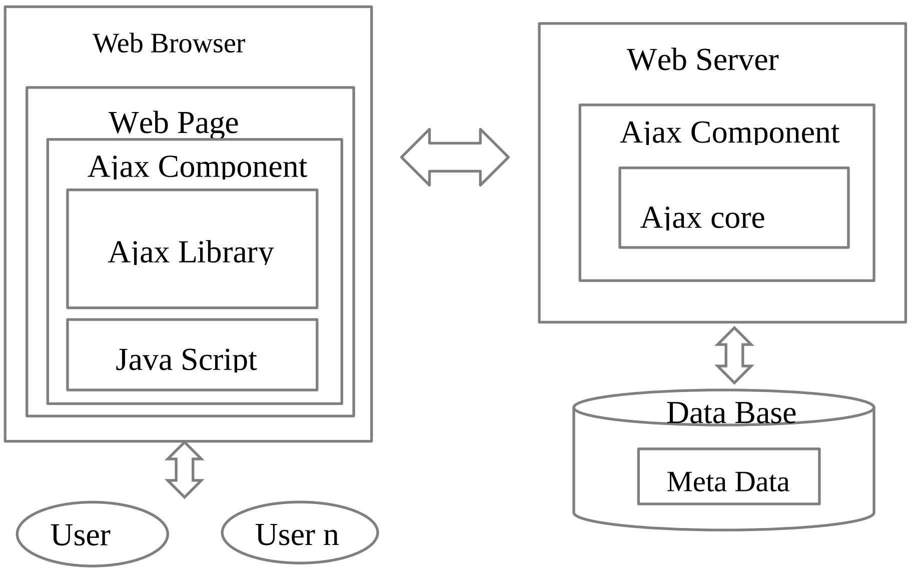

Fig. 1. Overall Structure of Ajax Component-based PushApplication

3.2 Ajax core

Ajax Core is a framework for push server development and defines key functions for realizing push service on server.

By providing this function as an abstract function, it is possible to easily realize push server by applying the function. Figure 2 shows the overall structure of the Ajax core inner classes.

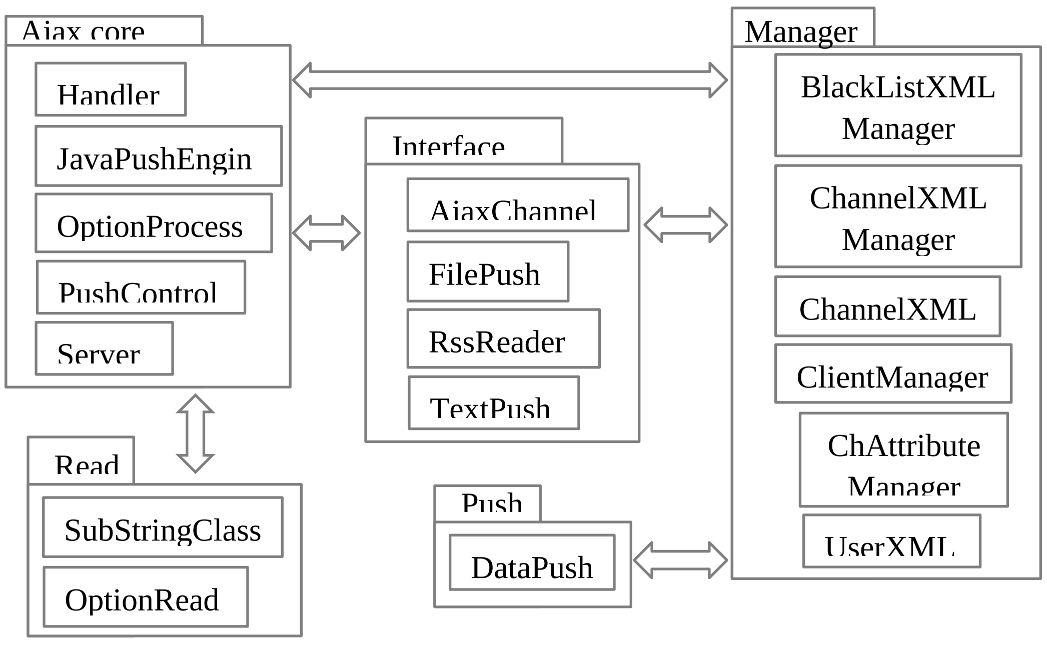

Fig. 2. Ajax core structure

The core package of Ajax core is responsible for setting the basic execution environment such as initialization and client connection to execute push service.

The process of push server execution and providing information to the client is as follows.

(1) Perform initialization for Push service while creating Server class object of Core package

(2) When the client connects and passes the notification message, the Handler class object forwards this message to the PushControl class object.

(3) The PushControl object retrieves the Channel through the ChAttbuteManager object and calls the Channel Manager object pointing to the Channel for Informational Purpose

(4) The ClientManager object is loaded from the ChannelManager object through the ChannelManager object, and the DataPush object of the Push package is used to provide information to the client.

The Ajax core provides a JavaPushEngine class for push services.

The JavaPushEngine class supports FilePush, RssReader, and TextPush objects to push various types of data internally, and provides them as an abstract function so that push server developers can easily develop push services.

Ajax core structurally manages channel or user related information through XML configuration file.

The ChannelList and User files record detailed information about the Channel and the user, respectively, and the Ajax core can use the ChannelAjax class to retrieve or modify the necessary information in this file.

The Ajax core uses a Channel to represent an effective information classification and client list.

The ChannelManager class provides access to hierarchical access to the list of channels stored in the channel list and the list of clients in the list.

Figure 3 shows a hierarchical data structure for channel management according to the Ajax component client.

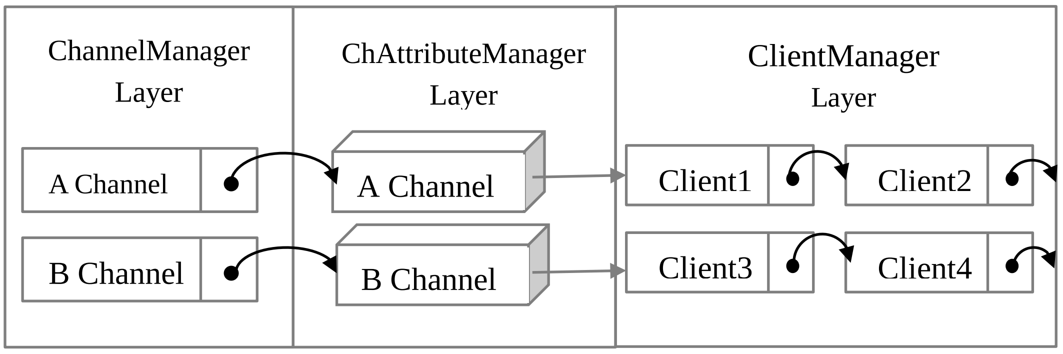

Fig. 3. Multi-Layer Structure of Channel for Management of Clients

The ChAttributeManager class in the middle layer stores properties that can efficiently utilize the various functions of the Channel. Table 3 shows the attributes and functions that make up the Channel.

Attributes	Functions
title	Channel name
max	Maximum number of connected clients on this channel
isopen	Channel Open / Close Settings
isgroup	Setting up group membership
type	Channel's visibility
password	Set private channel secret password
userXML	XML file referenced in personal user authentication service

Table 3. Attributes and Functions of Channel

The realization of the channel is made as an XML file that is independent of the Ajax component core, so that even if the server is running, the ChannelXML file can store the attributes of the current channel.

Code 1 shows a DTD (Document Type Definition) [24] file that defines the structure of the attributes that make up the Channel.

```
<!ELEMENT Channel (title, max, isOpen, isGroup, type, password, usersXML) >
<!ELEMENT title (#PCDATA) >
<!ELEMENT max (#PCDATA) >
<!ELEMENT isOpen (#PCDATA) >
<!ELEMENT isGroup (#PCDATA) >
<!ELEMENT type (#PCDATA) >
<!ELEMENT password (#PCDATA) >
<!ELEMENT usersXML (#PCDATA) >
```

Code. 1. Definition of Attribute Structure of Channel

To realize push function, we used Java NIO's SocketChannel class function to support client socket communication. Figure 5 shows the source code that calls the client on the Channel to provide information.

```
public void Push(ClientManager queue, ByteBuffer buf){
SocketChannel sc = null;
for(int s = queue.queueManager.size(); s>0; ss--){
sc = queue.queueManager.pop();
// If the client is a client connected to the server Push
if(sc.isConnected()){
sc.write(buf);
sc.close();
```

Fig. 5. Sample Code for Information Push

Call the SocketChannel object contained in the ClientManager object and check whether the client is connected to the push server through the is Connected () function. If the client is connected, send the notification message using the write () function.

3.3 Ajax library

The Ajax library is a framework for developing clients that receive push service requests and push information while communicating with push servers.

The Ajax library consists of a Config file that manages configuration information related to the push server and an Ajax client library that performs functions such as request and information reception while communicating with the push server.

Especially, Ajax client provides push client development environment by abstracting key functions such as push server access, push service request, channel setting and information reception.

The Ajax client library internally implements functions such as communication with the push server, push service request and information reception, and supports these functions by abstracting the JavaScript function.

The client can realize Push service based on this function.

The notification to be sent from the Ajax client to the push server consists of inChannel, outChannel, message property, and defines the connection channel, destination channel, and notification contents respectively.

These notifications are generated in JSON form in the CreateJSON () function at the request of the user and converted into a string.

The converted notification is then sent to the server through the XHR object in the ChannelEnter () function.

The Ajax client's Response () function handles notifications that are pushed from the server. The Response () function supports attributes such as display and disOption to support output from the Ajax library, and also supports the func property for output using the method specified by the user.

You can also handle XML notifications through the xmlDOC attribute.

The Ajax client uses the user () and Channel () functions to request the configuration of the main meta information of the push server. If the user sends information about the user and channel through the user () and channel () functions of the client, the push server verifies the validity of this information, stores the relevant information in an XML-formatted metafile, and uses it in the system.

```
    。。。。。。
    // Functions for request to push server
function ChannelEnter(msgJSON, func, xmlFunc,
display, disOption, inChannel){
myAjax = createAjax();
myAjax.open("post","http://"+ Ajax component IP,true);
myAjax.onreadystatechange = function call(){
// Function to be executed when responding
response(msgJSON, func, xmlFunc, display,
disOption, inChannel);
};
myAjax.send(msgJSON);
```

Fig. 7. Implementation of Asynchronous Communication using XHR

The realization of Ajax library consists of creating XHR object to perform asynchronous communication with push server and realization of function which takes this object.

Figure 7 shows part of the source code for XHR object creation for asynchronous communication and XHR object for server and asynchronous communication.

The createAjax () function creates an XHR object to perform asynchronous communication.

It requests the asynchronous communication connection to the designated push server through the next open () function. When it receives information from the server, it processes the information through the call () function specified in onreadystatechange attribute.

The call () function calls the response () function to analyze and process the received information.

Figure 8 shows the processing of the information provided by the response () function called from the call () function.

```
    // Run when you receive a response
function response(msgJSON,func,xmlFunc,
display,disOption,inChannel){
var able = 1;
if(myAjax.readyState == 4){
if(myAjax.status == 200 || myAjax.statusText ==
"OK"){
...
// Specify output format according to disOption
if(disOption=="value"){
display.value = myAjax.responseText;
}else if(disOption=="innerHTML"){
display.innerHTML = myAjax.responseText;
}else if(disOption=="alert"){
...
   // Reconnect to push server
ChannelEnter(msgJSON,func,xmlFunc,display,
disOption, inChannel);
```

Fig. 8. Process of Information Push in "response()"

The XHR object checks the validity of the push information with the readyState attribute and analyzes the push type from the disOption attribute.

Then, the output format is set according to the analyzed flow type. When the output is completed on the indicator, ChannelEnter () function is set up to receive the next push information by retrying asynchronous communication connection with the push server.

3.4 Ajax component communication protocol

Push server based on Ajax core and client based on Ajax library exchange request and response information according to Ajax component communication protocol.

The Ajax component communication protocol defines various attributes to effectively represent the information of the request and response, and all are realized based on the JSON object.

Table 4 shows the main attributes and functions of the JSON object defined in the Ajax component..

Attributes	functions
inChannel	Channel to which the client is trying to connect

outChannel	The Channel name to which the client will provide information.
message	Notification to Push
option	Attributes that set the request defined on the push server
optionMessage	A JSON object that supports the option attribute

Table 4. Standard Attributes of Ajax Communication Protocol

The option attribute requests a request to modify an XML file or a channel with a password to set the request defined in the AJAX component core.

Table 5 shows the functionality of the option attribute defined in the Ajax core.

Values	Functions
addChannel	Channel Add
delChannel	Channel delete
modChannel	Channel Modified
closeEnter	Connect to private channel
userLogin	Individual user access
addUser	User registration
delUser	Delete user

Table 5. Values and Functions of "option" Attribute

An Ajax client dynamically creates an optionMessage according to the option attribute to request different option functions.

optionMessage has a request notification to the push server based on the JSON object and shows the function of the option Message attribute in Table 6.

option type	Attribute	Explanation
addChannel delChannel modChannel	chName	Channel name
	maxUser	Maximum number of clients allowed to connect
	open	Visibility
	close	Make private
	chPw	Set password
	pw	Channel password
closeEnter userLogin addUser delUser	channel	Channel name
	id	User ID
	password	User password

Table 6. "optionMessage" Attribute along with "option"

5. conclusion

The paper describes the design and implementation of Ajax components for asynchronous communication in Web services.

The Ajax component consists of an Ajax core for the push server for asynchronous communication and an Ajax library for the push client that interacts with it. The Ajax core performs asynchronous communication with clients based on the Selector class of the Java NIO and handles many clients concurrently by supporting Linux Epoll technology.

In addition, it supports various execution environments by operating on Java-based, and it has various application ranges because it provides information classified by users through multiple channels.

Push server developers can easily utilize the push function and realize various push services using the functions provided in the JavaPushEngine class of Ajax component.

The Ajax library performs Ajax communication as a JavaScript client that performs asynchronous communication with a push server based on Ajax core part.

Ajax's XHR object maintains a persistent connection with the push server to achieve long polling push between the client and the server. XHR also uses JSON objects to effectively represent and transmit client information.

The JavaScript-style Ajax library operates in various browsers and supports a flexible push client development environment through abstraction of push services. Especially, Ajax component manages channel and user information as metafile, and provides various application creation interface which can easily access and configure it, so that it can easily realize the management function of Push app.

The developed Ajax component provides an application programming interface to realize push technology to support effective development of Web business service programs.

Ajax components can be widely used as a framework for Web business services applications that provide massive and constantly changing information efficiently.

REFERENCES

[1] Shishir Gundavaram, "CGI Programming on the World Wide Web," 1st Edition, O'Reilly Media, 1996.

[2] "http://en.wikipedia.org/wiki/Push_technology," Push Technology.

[3] Andy Zaidma , Nick Matthijssen. “Understanding Ajax applications by connecting client and ser ver-side execution traces” Empir Software Eng, vol 18,pp.181–218.2013

[4] Amalfitano D, Fasolino AR, Polcaro A,”Dynaria: a tool for Ajax Web application comprehension. In: Proceedings of the international conference on program comprehension”.IEEE Computer Soci et y, pp 46–47.2010

[5] Matthijssen N, Zaidman A “Firedetective: understanding Ajax client/server interactions. In:Procee dings of the international conference on software engineering” (ICSE). ACM, pp 998–1000. 2011

[6] Mesbah A, van Deursen A”A component- and push-based architectural style for Ajax applications”. J Syst Softw vol 81,no 12.pp 2194–2209.2008

[7] Sachin Deshpande, Wenjun Zeng, “HTTP streaming of JPEG2000 images,” in the Proceeding of ITCC'01,IEEE Comput. Soc, pp.15-19, 2001.

[8] A. Russell, G. Wilkins, and D. Davis. Bayeux – a JSON protocol for publish/subscribe event delivery protocol 0.1draft3. http://svn.xantus.org/shortbus/trunk/bayeux/bayeux.html, 2007.

[9] "http://en.wikepedia.org/wiki/XHR." XMLHttpRequest

[10] "http://en.wikepedia.org/wiki/JSON." JSON