WEBIM的简单实现

Authors: bm888888@gmail.com

Abstract: 本文描述了使用javascript直接连接xmpp server的方法和详细配置，使用XEP-0124和xmpp server无缝接合。从而实现webim的实时性和高效性。

实现原理

实现webim方法有很多，最关键的问题是保持和服务端的连接。如何保障会话的即时性和连通性。常用的有poll, long poll, comet等;; 其中poll的方式效果最差，一般都会有1到2秒的延迟。long poll和comet技术比较新，因为http无状态的原因，这种长连接方式要保持，一般都需要服务端额外代码提供支持。像gtalk采用的就是long poll的方式。服务端常会使用异步IO等方式来支持高并发。

本文使用的是XEP标准扩展规范,XEP-0124 提供的方法实现WebIM. 思路即使用一个javascript的xep-0124的实现，直接连接xmpp server端。目前xep-0124的实现，在大部分的xmpp server实现中都支持，本文使用的是OpenFire 3.6.4 作为Xmpp Server。

XEP 0124规范定义了一个比较完整和安全的协议，具体请参阅相当文档。本文使用javascript端的XEP-0124的实现为Strophe的js实现。

另外因为浏览器javascript跨域问题，需要使webim里调用javascript也是80端口，并同一子域，所以使用Apache或者Nginx 在80端口，并转发/http-bind请求至Xmpp Server的http-binding端口。本机使用Nginx. xmpp server 使用使用7070端口。 结构为：

```
Web UI page.html
     ||
     ||  strophe.js Ajax request /http-bind/
     ||
     >
 Nginx/Apache   port: 80   /http-bind/
     ||
     ||
     >
  XMPP Server http-bind port (OF: 7070) /http-bind/
```

安装准备

下载OpenFire 3.6.4并安装。http://www.igniterealtime.org/downloads/index.jsp
下载Strophe的javascript库 http://code.stanziq.com/cgit/strophe/strophejs/snapshot/strophejs-master.tar.gz
下载Nginx或者Apache并安装配置。只需一个，作为80端口服务的代理转发服务器。

安装配置

OpenFire的配置

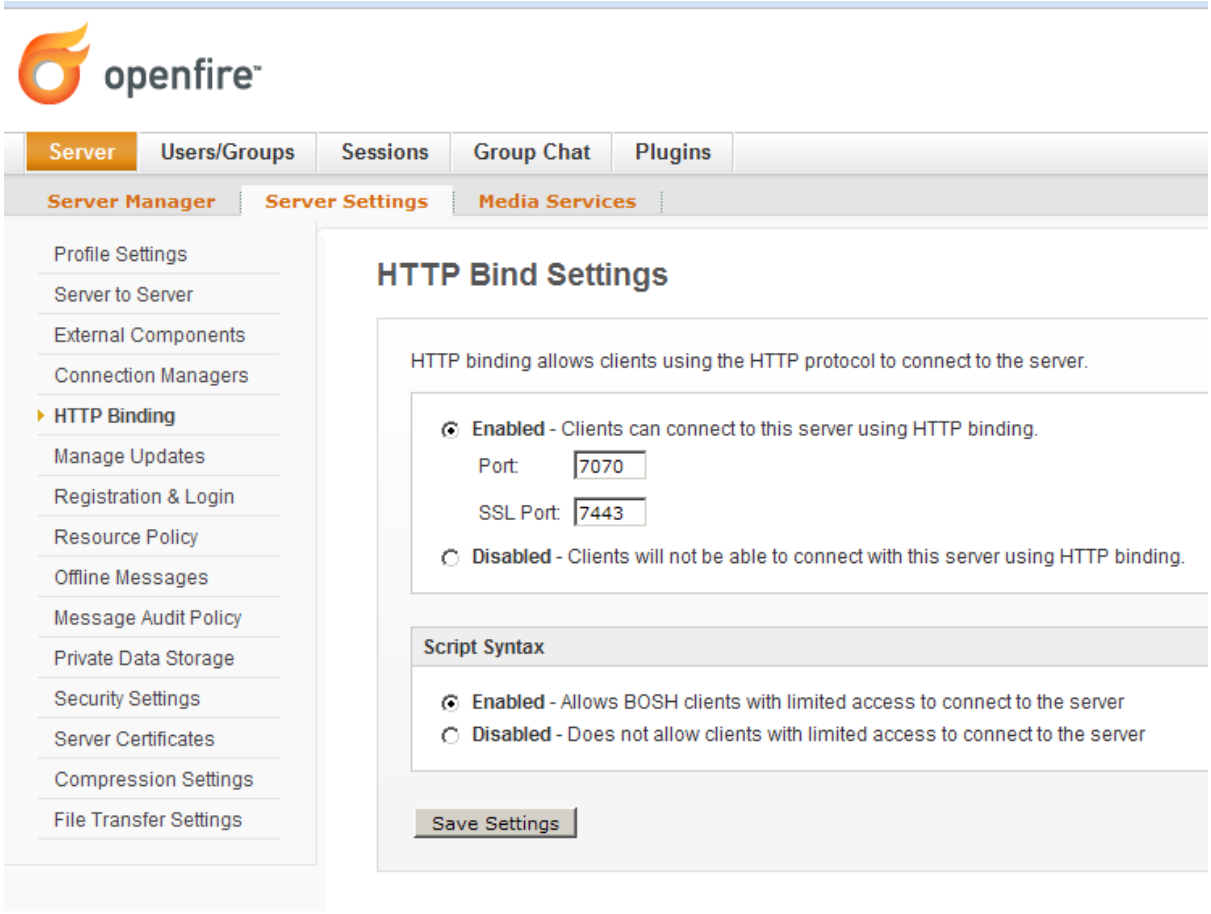
需要打开OpenFire的http-binding.具体为登录后台管理界面。Server->Server Settings->Http Binding:

这一项打勾 **Enabled** - Clients can connect to this server using HTTP binding.

下面**Script Syntax** ：

这一项打勾 **Enabled** - Allows BOSH clients with limited access to connect to the server

端口按默认的。



Nginx的配置

打开nginx的配置文件nginx.conf，增加如下行：

```
#gzip  on;

#增加
  upstream bk.openfire {
      server 127.0.0.1:7070;
   }
```

```
location / {
        root   html;
        index  index.html index.htm;
}

后面增加 ：

location /http-bind {
        proxy_pass http://bk.openfire;
        proxy_buffering off;
        proxy_redirect off;
        proxy_read_timeout 120;
        proxy_connect_timeout 120;
}
```

黑字为原来的配置， 红色为需要增加的行。

Apache的配置

在 httpd.conf 中加入下面几行：

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so


ProxyRequests Off
ProxyPass /http-bind http://127.0.0.1:7070/http-bind/
ProxyPassReverse /http-bind http://127.0.0.1:7070/http-bind/
```

注：需要apache有编译proxy模块。

编写WebIM

上面就把服务器配置好了，下面我们来开发我们的简易版的纯WEB的IM.

解压strophejs-master.tar.gz，后，可以得到 b64.js , md5.js sha1.js strophe.js ，这是我们要用到的几个js基础类库。在我们的页面中将引用。里面自带了一个examples目录，也可以学习里面的例子，以熟悉strophe的语法。

我这个例子，就从echobot.html这个example里修改得到。

具体的UI布局部分省略，都是基本的html元素加上几个样式，看代码就清楚。

Strophe是一个优秀的xep-0124类库，使用它非常方便地就集成了。

Strophe建立连接

以下代码在echobot.js中。

代码1：使用Strophe建立服务端连接

```
function onConnect(status)
{
    if (status == Strophe.Status.CONNECTING) {
log('Strophe is connecting.');
    } else if (status == Strophe.Status.CONNFAIL) {
log('Strophe failed to connect.');
$('#connect').get(0).value = 'connect';
    } else if (status == Strophe.Status.DISCONNECTING) {
log('Strophe is disconnecting.');
    } else if (status == Strophe.Status.DISCONNECTED) {
log('Strophe is disconnected.');
$('#connect').get(0).value = 'connect';
    } else if (status == Strophe.Status.CONNECTED) {
log('Strophe is connected.');

connection.addHandler(onMessage, null, 'message', null, null,  null);
connection.send($pres().tree());
    }
}
```

```
connection = new Strophe.Connection(BOSH_SERVICE);

    connection.rawInput = rawInput;
    connection.rawOutput = rawOutput;

    //Strophe.log = function (level, msg) { log('LOG: ' + msg); };


    $('#connect').bind('click', function () {
var button = $('#connect').get(0);
if (button.value == 'connect') {
   button.value = 'disconnect';

        fromId = $('#jid').val();
        toId = $('#tojid').val();

        log(fromId);
        log(toId);

   connection.connect($('#jid').get(0).value,
      $('#pass').get(0).value,
      onConnect);
} else {
   button.value = 'connect';
   connection.disconnect();
}
```

```
    });
```

以上代码说明：

BOSH_SERVICE ： 这是xmpp server 的http binding的url地址。注:Openfire为: http://127.0.0.1:7070/http-bind/ 我们需要使用Apache或者nginx作Proxy转发，所以这里配置是 "/http-bind/" 。

使用Connection.connect方法连接OF服务器，传入用户名、密码、以及连接成功的回调函数onConnect三个参数。

connection.addHandler(onMessage, null, 'message', null, null, null);

这是连接成功后，增加一个处理消息的回调函数。当收到消息时，会调用onMessage函数。

代码2：onMessage()函数

```
function onMessage(msg) {
    to = msg.getAttribute('from');
    var from = msg.getAttribute('from');
    var type = msg.getAttribute('type');
    var elems = msg.getElementsByTagName('body');

    if (type == "chat" && elems.length > 0) {
   var body = elems[0];
     appendToHis(new Date().toLocaleTimeString() + '  ' + from + ' say: ' +
Strophe.getText(body));
    }

    // we must return true to keep the handler alive.
    // returning false would remove it after it finishes.
    return true;
}
```

这里，msg是收到的消息，使用Strophe.getText(body) 这行，返回了收到的IM消息。
msg里还有一些其它的内容，如果你关心，可以接着处理它。

Strophe发送消息

```
msg=$('#msg').val();

toId = $('#tojid').val();
var reply = $msg({to: toId, from: fromId , type:
'chat'}).cnode(Strophe.xmlElement('body', '' ,msg));
connection.send(reply.tree());
```

也非常简单，先组装好一个消息，然后调用
调用connection.send() 发送消息即可。

Strophe关闭连接

```
connection.disconnect();
```

Strophe的日志和调试

你可以通过编写一行日志处理函数，来跟踪strophe.

把这行注释去掉，并在函数实现里写你的日志处理就可以。

```
//Strophe.log = function (level, msg) { log('LOG: ' + msg); };
```

本例子中用的log()函数：

```
function log(msg)
{
    $('#log').append('<div></div>').append(document.createTextNode(msg));
}
```

直接把内容输出到本页的一个log div里，注：使用的是jquery的语法。

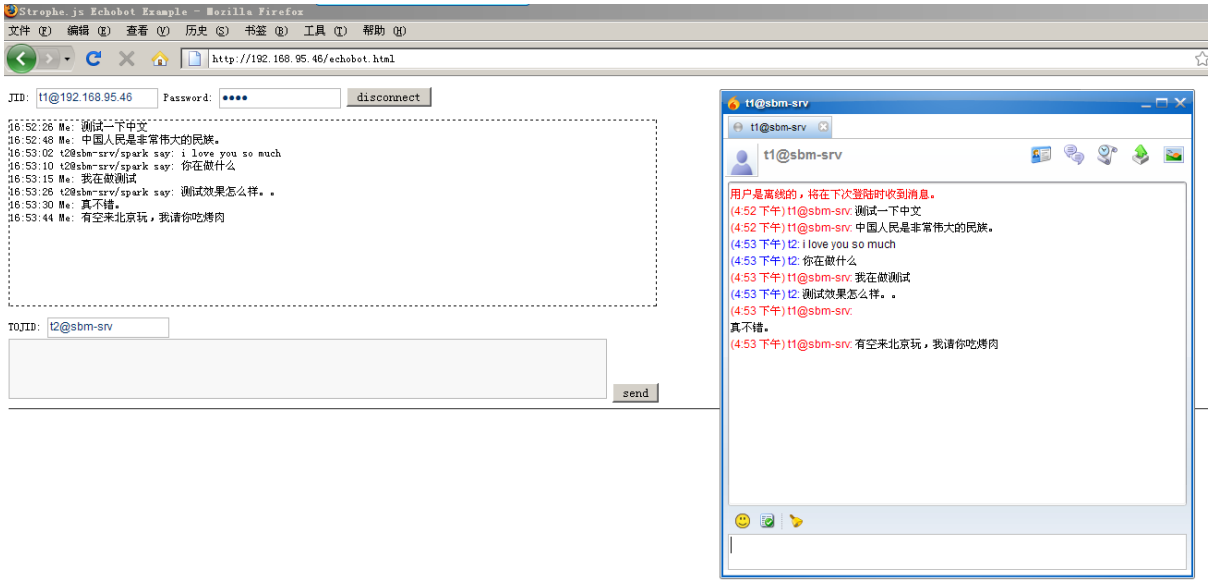
另外你还可以定义rawInput,rawOutput函数来监控connection上的IO内容，可以看例子中的代码。

```
    connection.rawInput = rawInput;
    connection.rawOutput = rawOutput;

...
function rawInput(data)
{
    log('RECV: ' + data);
}

function rawOutput(data)
{
    log('SENT: ' + data);
}
```

本例子中，把发送和接收的内容，也输出到log中。

另外，你还可以使用Firefox的firebug插件来调试Javascript。

运行效果



本文完整**Source**下载

//TODO
GOOGLE DOCS 不提供上传文件功能？我得找个地址。。。

webimclient.zip

参考手册

[1] XEP 0124 http://xmpp.org/extensions/xep-0124.html
[2] The homepage for Strophe is http://code.stanziq.com/strophe
[3] Strophe 的API Doc http://people.chesspark.com/~jack/strophe-preview/doc/files/strophe-js.html