

The following guide shows how to write a simple chat client implemented in [JavaScript](#) and [PHP](#).

Contents

- [1 How it works](#)
 - ◆ [1.1 JavaScript](#)
 - ◆ [1.2 PHP](#)
- [2 Requirements](#)
- [3 Future work](#)
- [4 Follow up](#)
- [5 Author](#)

How it works

[HTTP](#) is not the ideal protocol for a Chat. If you want to have a real chat server you should go for [IRC](#) or [Jabber](#).

While not overcoming the limitations of a stateless protocol, the [AJAX](#) technology at least made it possible to implement a web chat that does not bring down the server, although a real mobile chat has to be implemented to be as resource efficient as possible.

The current chat have to be extended to maintain the operations on the server less complex and only sent updated data to the clients to keep the traffic low.



JavaScript

The code bellow shows how to make a call to an external server.

```
function sync(aMsg) {
```

How_to_create_a_chat_in_WRT

```
url = "http://localhost/forumnokia/forumnokia.php?msg=" + aMsg;
msg=msg;
ext_doc.XMLHttpRequest();
ext_doc.onreadystatechange = onMsgLoad;

try {
    ext_doc.open("GET", url, true);
    ext_doc.send(null);
} catch(e) {
    alert(e);
}

function onMsgLoad() {
    if(text_doc.readyState == 4|| text_doc.readyState == "complete") {
        divMessage=document.getElementById("divChatMessaging");
        divMessage.innerHTML = ext_doc.responseText;
    }
}
```

PHP

The following PHP file was used in this sample to hold the server side logic in an unique resource.

```
<?
function add_message($msg, $log_dir, $full_path) {
if(file_exists(($log_dir . '/'))) {
$f = fopen($full_path, "a");
$dt = date("Y-m-d h:i:s");
$msg = urlencode(strip_tags(stripslashes($msg)));
$msg = "$dt $msg\n";
$remote = $_SERVER["REMOTE_ADDR"];

fwrite($f, $msg);
fclose($f);

return $msg;
} else {
@mkdir("log", 0700);
($msg);      add_message
}
}

function load_message($msg, $full_path) {
$content_array = file($full_path);
$saved = array_slice($content_array, -20);
$result = '';

foreach ($saved as $value) {
$result .= $value . '<br />';
}

if($msg != '1')
$result = substr($result, strpos($result, $msg) + strlen($msg) + 7, strlen($result));
if(strlen($result) == 0)
$result = 'nodata';

return $result;
}
```

How_to_create_a_chat_in_WRT

```
$action = $_REQUEST['act'];
$msg = $_REQUEST['msg'];

$log_dir = 'log';
$log_file = 'wall.html';
$full_path = $log_dir . '/' . $log_file;

if(isset($msg)) {
if($action == 'add') {
echo add_message($msg, $log_dir, $full_path);
} else {
echo load_message($msg, $full_path);
}
}
?>
```

Requirements

The widget logic is written in JavaScript and through the XMLHttpRequest API we request data from a server written in PHP. The chat messages are stored in a text file just to turn things less complex.

- Server side
 - ◆ PHP >=4
- Mobile side
 - ◆ Widget Support ([check the complete list of supported devices](#))

Future work

To complement this tutorial you can use a Flash Lite interface to add sound support and/or to establish the (optional) socket connection on client side.

There is an [article](#) at [Adobe Mobile and Devices Developer Center](#) that takes you through XML Sockets connections in Flash Lite 2.1 and higher and provides a hands-on demonstration of how to create mobile Flash Lite multiplayer content with a Java NIO Server.

Follow up

Please, [[contact me](#)] to get the full source code. There are many resources available that you can use to be more confident with widgets development. Check bellow some of them:

[Widgets for the S60 Platform eLearning](#)

[Getting Started with Web Runtime Widgets for S60 Screencast](#)

[Web Developer's Library](#)

Author

--FelipeAndrade 20:10, 12 September 2008 (EEST) [Profile](#)