HTML5 Based Ajax and SSE Technology with Server Intelligent Instant Messaging to Meet the Needs of Transactional Websites

Shuangyuan Li

Jilin Institute of Chemical Technology, 45 Chengde Street, Longtan District, Jilin City, Jilin Province, China

Abstract. With the continuous development of science and technology, Internet technology has been quietly affect the mode of transmission, traditional web site information data only through the web page to load when it is open, need to update the web page information can only be accomplished by way of a page refresh. In today's efficient informatization environment, has highlighted its problems, restricted information updated timely. In this paper, the new Internet programming language of HTML5 is combined with Ajax and SSE technology to study intelligent instant messaging and realize the new application mode.

Keywords: html5; Ajax.; SSE.

Nowadays, the Internet technology covers a wide range of applications, and the advantage of Html5 technology is used to create new application models and bring better experience to users, which is the focus of current research. As a designer, we should consider from the perspective of user demand, pay attention to user experience and meet personalized needs. For example, the stock information website, visitors can get the latest stock market information by refreshing the page, and cannot update the data timely and efficiently. For example, all kinds of e-commerce promotional activities, because the page reload time after the missed a transaction. This waste of time and server resources cannot be ignored, so the immediacy of web page data transmission is becoming increasingly important.

HTML5 is a set of technologies including HTML, CSS and JavaScript that enhance the performance of Web pages and add the functions of Web applications such as local databases. Ajax is called "Asynchronous Javascript And XML" (Asynchronous Javascript And XML). It refers to a web development technology that creates interactive web applications. By exchanging small amounts of data with the server in the background, Ajax can make web pages achieve Asynchronous update. Server-sent Event, or SSE for short, is a component of the HTML 5 specification that can be used to push data from the Server to the browser in real time. By combining the above technologies, intelligent instant messaging can be realized to meet users' demand of real-time information acquisition.

1 Interactive page design

First, design a simple page for interactive demonstration.

```
<form action="http://127.0.0.1/test.php?action=ajax" method="post" id="action">
<span>username</span><input type="text" name="usr_id" value="" id="txt1">
<span>password</span><input type="text" name="usr_psw" value="" id="txt2">
<input type="submit" name="usr_btn" value="submit" id="sub1">
<pre id="pre_sse"></pre>
    <pre id="pre_ajax"></pre>
    </form>
```

Given the space limitations, the basic tags of the web page are omitted and only the tags set for the implementation function are retained. The <form> tag is used to realize the manual interaction between the web page and the server, and two <pre> tags are reserved to display the data obtained through ajax and SSE. Here, the action address of the <form> tag fills in the server program that is ready to handle the ajax information.

2 Design of data transmission module

Write test.php on the server side to handle ajax transfer data and SSE.

```
<php?
    switch ($_GET['action']) {
    case 'ajax':
    echo 'ajax return to success';
    break;
case 'sse':
    header('Content-Type: text/event-stream');
        header('Cache-Control: no-cache');
```

Shuangyuan Li: lsy@jlict.edu.cn

```
        $time = date('Y-m-d H:i:s');
        echo 'retry: 1000'.PHP_EOL;
        echo 'data: now is: '.$time.PHP_EOL.PHP_EOL;
            break;}
?>
```

The code above implements the classification of ajax and sse requests through action values of get. The "ajax request" data is returned when the ajax request is obtained from the web page, and a "data:now is server current time" data is returned every 1000ms when the sse request is obtained.

3 Design of data transmission and processing module

In order to realize the sending and processing of ajax and sse data, a test.js was written to be introduced into the web page. Meanwhile, jquery library was introduced into the web page.

```
$(document).ready(function() {
vares=new EventSource("http://127.0.0.1/test.php?action=sse");
es.addEventListener("message",function(e){
            //e.data
            $("#pre_sse").html(e.data);
        },false);
        $('#action').submit(function(a) {
            $url1=$("#action").attr("action");
a.preventDefault();
$.post($url1,{usr_id:$('#txt1').val(),usr_psw:$('#txt2').val(),usr_btn:$('#sub1').val()},function(data,status){
if(status=='success') {$sat='ajax Successful data exchange';}else{$sat='ajax Data exchange failed';}
$("#pre_ajax").html($("#pre_ajax").html()+"</br>"+$sat);  }); }); });
```

Then the basic module of the required function is completed. When you open the prepared web page with the browser, you can see the login interface as shown in Figure 1:

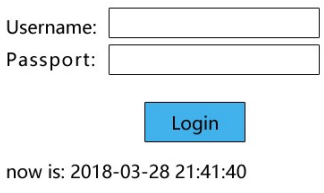

Figure 1. Login interface

And now is: the later time is always changing with the change of time in the server environment. Enter any user name or password box, click login, and there will be an additional line under the time:

```
Ajax successful data exchange
Ajax successful data exchange
Ajax successful data exchange
Ajax successful data exchange
Ajax successful data exchange
```

This enables real-time interaction with the server through Ajax and SSE as well as manual interaction with the data.At this point, visitors can update the required information in real time without refreshing the page.

4 Optimal Design

If complete above operation only, its intelligence is reduced greatly. The Php file can be modified as follows for special cases where the data required by the user does not change and there is no need to re-send the useless data to replace the original data.

```
<?php
session_start();
if (isset($_SESSION['value'])){
    $first=false;
}   else{
    $first=true;}
    switch ($_GET['action']) {
        case 'ajax':
            echo 'ajax return to success';
            break;
        case 'sse':
        header('Content-Type: text/event-stream');
        header('Cache-Control: no-cache');
        $time = date('Y-m-d H:i:s');
        echo 'retry: 1000'.PHP_EOL;
        if ($first) {
            $_SESSION["value"]=$time;
            echo 'data: now is: '.$_SESSION["value"].PHP_EOL.PHP_EOL;
        }   else{
            if ($_SESSION['value']==$time) {break;}
    else{$_SESSION['value']=$time;
                echo 'data: now is: '.$_SESSION["value"].PHP_EOL.PHP_EOL;}}
            break;}
?>
```

Through the above code is managed to user data needed for temporary storage by the session on the server, every time when the trigger SSE request, the server will authenticate users need $time data is the data with the last changed, when there is no change, then you won't send change data to the client to repeat, to avoid the client due to rewrite data additional CPU and memory usage, and both the optimization of resource usage is especially important in mobile terminal. Since the $time data set is generated by the current time, this determines that it is always updated per second, so this feature is not represented. In practice, it can be replaced by other required data.

Users continues to increase, although is enabled on the server session variable, but still can cause the server memory footprint is cumulative, and the default session variables automatically destroyed time for 24 min, in the case of 1 s a SSE request frequency, obviously too long, so want to rewrite the survival time of the session variable, and the memory used by the session released when.

The specific modified code is as follows:

```
<?php
session_start();
if (isset($_SESSION['value'])){
    $first=false;
}   else{
    $first=true;}
    switch ($_GET['action']) {
        case 'ajax':
            echo 'ajax return to success';
            break;
        case 'sse':
        $later=1000;// Every 1000 is 1s, in ms
        header('Content-Type: text/event-stream');
        header('Cache-Control: no-cache');
        setcookie(session_name(), session_id(), time()
+ $later/1000*60, "/");
        $time = date('Y-m-d H:i:s');
        echo 'retry: '.$later.PHP_EOL;
        if ($first) {
            $_SESSION["value"]=$time;
            echo 'data: now is:
'.$_SESSION["value"].PHP_EOL.PHP_EOL;
        }   else{
            if ($_SESSION['value']==$time) {break;}
    else{
            $_SESSION['value']=$time;
                echo 'data: now is:
'.$_SESSION["value"].PHP_EOL.PHP_EOL;}}
            break;}?>
```

With the above code Settings, the session storage time is changed to 60 times the SSE refresh cycle, which in the example is 60s(60000ms). In the actual application, users may need to temporarily suspend and continue to monitor the changes in the data during use. Therefore, HTML, js and PHP files need to be modified. The specific implementation method is as follows:

Html

```
<form    action="http://127.0.0.1/test.php?action=ajax"
method="post" id="action">
        <span>username</span><input    type="text"
name="usr_id" value="" id="txt1">
        <span>passport </span><input type="text"
name="usr_psw" value="" id="txt2">
        <input    type="submit"    name="usr_btn"
value="submit" id="sub1">
        <a  id="pause"  href="">  Suspension  of
communication </a>
        <pre id="pre_sse"></pre>
        <pre id="pre_ajax"></pre>
    </form>
Js:
$(document).ready(function() {
tmp=true;
functionSsestart(){
vares                =                new
EventSource("http://127.0.0.1/test.php?action=sse");
es.addEventListener("message",function(e){
            //e.data
        $("#pre_sse").html(e.data); },false);   };
Ssestart();
        $('#action').submit(function(a) {
            $url1=$("#action").attr("action");
a.preventDefault();
$.post($url1,{usr_id:$('#txt1').val(),usr_psw:$('#txt2').val
(),usr_btn:$('#sub1').val()},function(data,status){
if(status=='success')
                {$sat='ajax  Successful  data
exchange';}else{$sat='ajax Data exchange failed';}
$("#pre_ajax").html($("#pre_ajax").html()+"</br>"+$sa
t);});   });
        $('#pause').click(function(a) {
a.preventDefault();
if (tmp) {
$.post('http://127.0.0.1/test.php?action=pause',{action:'p
ause'},function(data,status){
if(status=='success'){
alert(data);   }});
tmp=false;
  $('#pause').html(' Get communication ');
}else    {
$.post('http://127.0.0.1//test.php?action=continue',{actio
n:'pause'},function(data,status){
if(status=='success'){
alert(data);   }});
tmp=true;
                $('#pause').html('  Suspension  of
communication ');
Ssestart();}});});
Php:
<?php
session_start();
if (isset($_SESSION['value'])){
    $first=false;}   else{
    $first=true;
    $_SESSION['sse_pause']=false;}
    switch ($_GET['action']) {
        case 'ajax':
            echo 'ajax Return to success';
            break;
        case 'sse':
        if(!$_SESSION['sse_pause']){
            $later=1000;// Every 1000 is 1s, in ms
header('Content-Type: text/event-stream');
        header('Cache-Control: no-cache');
        setcookie(session_name(), session_id(), time()
+ $later/1000*60, "/");
        $time = date('Y-m-d H:i:s');
        echo 'retry: '.$later.PHP_EOL;
    if ($first) {
            $_SESSION["value"]=$time;
            echo 'data: now is:
'.$_SESSION["value"].PHP_EOL.PHP_EOL;
        }   else{
            if ($_SESSION['value']==$time) {break;}
    else{$_SESSION['value']=$time;
                echo 'data: now is:
'.$_SESSION["value"].PHP_EOL.PHP_EOL;}}
        }else{
$_SESSION['sse_pause']=false;}
            break;
            case 'pause':
    $_SESSION['sse_pause']=true;
            echo ' Successfully suspended ';
```

```
            break;
            case 'continue':
            $_SESSION['sse_pause']=false;
                echo ' Successfully continued ';
        break;}
?>
```

Then, we can switch the pause and continue state of SSE through the hyperlink click event, so as to further define the user's needs. Then, we can make the hyperlink look like a button with CSS. The final page is shown in figure 2:

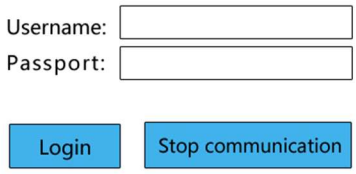

Figure 2. The login screen after compilation

5 Concludes

Through Ajax technology and SSE, instant messaging of the server is realized, which reduces the burden of the server. SEE can be used to push data from the server side to the browser side in real time, which is easier to operate and has less server side changes. Ajax can shift the work of the previous server burden to the client, use the idle capacity of the client to process, reduce the burden of the server and bandwidth, and minimize the burden of redundant requests and responses on the server, thus realizing the new application mode of intelligent instant messaging.

References

1. J.C. Xu, D.Y. Kong, Y.Y. Luo, Learning Support Technology of Mobile Online Education Platform Based on HTML5, College of Computer &Information Engineering, Henan Normal University Journal Natural Science Edition, **03**, 143-147 (2015).
2. R.Y. Rui, D.M. Zhang, The Research and Design of the Information System Based on Ajax and MVC Pattern, College of Electronic information, Electronic Technology Application Journal,**02**, 128-131 (2014).
3. S.J. Ren, X. Zhou, Y.M. Ren, L.L. Li, Multi-Thread Downloading Technology Based on HTML5, Journal of Computer System Application, **26(11)**, 11-18 (2017).
4. L. Yang, Research on Algorithm of Crawling Ajax Dynamic Web Pages Based on User Interface State Changes, Beijing Jiaotong University (2016).