Subject: Server Sent Events Example
Posted by omari on Sun, 21 Apr 2024 18:03:29 GMT
View Forum Message <> Reply to Message

Server-Sent Events (SSE) is a technology that enables a server to push real-time updates to a web application over an HTTP connection. Unlike traditional request-response mechanisms where the client initiates communication, SSE allows the server to initiate data transmission to the client once the client establishes a connection.

main.cpp:

```
#include <Core/Core.h>

using namespace Upp;

struct MySocket {
 TcpSocket socket;
 bool initialized = false;
};
class SSEServer
{
 private:
  TcpSocket server;
  Array<MySocket> clients;
  Mutex mx;

  bool end = false;
  Thread th;

  int port;

  bool accept()
  {
   if(end) return false;

   mx.Enter();
   clients.Add();
   mx.Leave();
   MySocket& client = clients.Top();
   bool ret = false;
   DUMP(clients.GetCount());
   if(client.socket.Accept(server)) {

    HttpHeader h;
    h.Read(client.socket);

    bool authorized = true;
    //  authenticate the user using h.f2 (/token)
```

```
   if(authorized) {
    client.socket.Put("HTTP/1.1 200 OK\r\n"                //Standard HTTP header line
         "Content-Type: text/event-stream\r\n"          //This is the only allowed MIME type for SSE
                "Transfer-Encoding: chunked\r\n"              //Chunked encoding lets it know when an
event is done without knowing packet boundaries.
       "Access-Control-Allow-Origin: *\r\n"           //Because the HTML comes from a file, not this
server, we have to allow access
         "\r\n");

         ret = true;
         client.initialized = true;
   }
   else {
    client.socket.Put("HTTP/1.1 401 Unauthorized\r\n"                //Standard HTTP header line
         "\r\n");
        ret = false;
   }
  }

  if(ret == false) {
   mx.Enter();
   clients.Remove(clients.GetCount() - 1);
   mx.Leave();
  }

  return ret;                                //End of header indicator
 }

 void WriteChunk(const String& str)
 {

  String out;
  out << FormatIntHex(str.GetCount()) << "\r\n"  << str << "\r\n";
  mx.Enter();
  for(auto& c:clients) {
   if(!c.socket.IsEof() && !c.socket.IsError()) {
    c.socket.Put(out);
   }
  }

  for(int i = clients.GetCount(); i > 0 ; i--) {
   MySocket& c = clients[ i - 1 ];

   if(c.initialized && ( c.socket.IsEof() || c.socket.IsError())) {
    clients.Remove(i-1);
   }
  }
```

```
  mx.Leave();
 }

public:

 SSEServer(int Port = 6500): port(Port)
 {
  server.Listen(port, 5);
 }

 ~SSEServer()
 {
  mx.Enter();
  for(auto&client: clients) client.socket.Close();
  mx.Leave();
  server.Close();
 }

 void Start() {
  Cout() << "Listn at port " << port << "\n";
  end = false;
  th.Run([&] {while(!end){accept(); Sleep(1);}});
 }

 void Stop() {
  end = true;
  TcpSocket soc;
  soc.Connect("localhost", port);
  soc.Connect("localhost", port);
  soc.Connect("localhost", port);
  th.Wait();
 }

 void WriteData(String d) { WriteData(~d); }
 void WriteData(const char* data) {
  //Build up an event in server-send-event format. The message consists of
  //one or more fields of the form:
  //field: value\n
  //Followed by an empty line.
  //
  //The main tag is "data:" which carries the data payload.
  //See
https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events/Using_server-sent_events
  //for more info (e.g. different message types and dispatch)
  String msg;

  if(Upp::Random(10)>5)
   msg << "event: " << "custom" << "\n"; // Send same msgs as 'custom' event instead  of
```

'message' (the default).

```
   msg << "data: " << data << "\r\n";
   msg << "\r\n";  //Empty field ends the message.

   //Send the message data using chunked encoding
   WriteChunk(msg);

  }
};

CONSOLE_APP_MAIN
{
 SSEServer sse;
 bool end = false;
 Thread th;
 th.Start([&] {
  while(!end){
  Time t = GetSysTime();
  String s ;
  s << t;
  sse.WriteData(s);
  Sleep(1000);
  }
 });

 sse.Start();
 Cout() << "Press enter to terminate\n" ;
 ReadStdIn();
 end = true;
 sse.Stop();
 th.Wait();
}
```

index.html:

```
<!DOCTYPE html>
<html>
<head>  <meta charset="UTF-8"> </head>
<body>

<h1>Getting server updates</h1>
<div id="result"></div>

<script>
if(typeof(EventSource) !== "undefined") {
```

```
    var source = new EventSource("http://127.0.0.1:6500/token");
    source.onmessage = function(event) {
        document.getElementById("result").innerHTML += "Data: " + event.data + "<br>";
  // console.log(event);
    };

    source.addEventListener("custom", function(event) {
        document.getElementById("result").innerHTML += "Custom: " + event.data + "<br>";
  // console.log(event);
    });

    source.onerror = function(event) {
        document.getElementById("result").innerHTML += "Connection failed<br>";
    };
} else {
    document.getElementById("result").innerHTML = "Sorry, your browser does not support
server-sent events...";
}
</script>

</body>
</html>
```