CURSO DESAROLLO WEB CON MEAN

WEB FULL STACK DEVELOPER

Germán Caballero Rodríguez
germanux@gmail.com

Conexiones TCP en tiempo real con

socket.io

INDICE

Sockets

- El protocolo HTTP fue concebido desde sus orígenes para ofrecer comunicaciones en un sólo sentido, desde el servidor hacia el cliente.
- Sin embargo las aplicaciones web de hoy en día demandan más que eso para poder ofrecer una experiencia de usuario más rica, necesitan flujo de información en ambos sentidos en el mismo instante en el que ocurren los eventos.

Sockets

- Socket designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.
- El término socket es también usado como el nombre de una interfaz de programación de aplicaciones (API) para la familia de protocolos de Internet TCP/IP, provista usualmente por el sistema operativo.

Sockets

- Los sockets de Internet constituyen el mecanismo para la entrega de paquetes de datos provenientes de la tarjeta de red a los procesos o hilos apropiados.
- Un socket queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto.

Sockets

Explicación detallada [editar]

Para que dos programas puedan comunicarse entre sí es necesario que se cumplan ciertos requisitos:

- Que un programa sea capaz de localizar al otro.
- Que ambos programas sean capaces de intercambiarse cualquier secuencia de octetos, es decir, datos relevantes a su finalidad.

Para ello son necesarios los dos recursos que originan el concepto de *socket*:

- Un par de direcciones del protocolo de red (dirección IP, si se utiliza el protocolo TCP/IP), que identifican la computadora de origen y la remota.
- Un par de números de puerto, que identifican a un programa dentro de cada computadora.

Los *sockets* permiten implementar una arquitectura cliente-servidor. La comunicación debe ser iniciada por uno de los programas que se denomina programa "cliente". El segundo programa espera a que otro inicie la comunicación, por este motivo se denomina programa "servidor".

Un *socket* es un proceso o hilo existente en la máquina cliente y en la máquina servidora, que sirve en última instancia para que el programa servidor y el cliente lean y escriban la información. Esta información será la transmitida por las diferentes capas de red.

Web Sockets en HTML5

- Con la aparición de HTML5 y nuevas características apareció la tecnología WebSockets basada en el protocolo ws.
- Se creó para mantener una conexión continua entre cliente y servidor por lo que la comunicación se hace más fluida que con las tradicionales llamadas http.
- Esto pinta un escenario ideal para aplicaciones chats, juegos en líneas o mostrar información en tiempo real.

Web Sockets en HTML5

- WebSocket es una tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP.
- Está diseñada para ser implementada en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor.
- La API de WebSocket está siendo normalizada por el W3C, mientras que el protocolo WebSocket ya fue normalizado por la IETF como el RFC 6455.

Web Sockets en HTML5

- Debido a que las conexiones TCP comunes sobre puertos diferentes al 80 son **habitualmente bloqueadas** por los administradores de redes, el uso de esta tecnología proporcionaría una solución a este tipo de limitaciones proveyendo una funcionalidad similar a la apertura de **varias conexiones** en distintos puertos, pero **multiplexando** diferentes servicios WebSocket sobre un único puerto TCP (a costa de una pequeña sobrecarga del protocolo)..

Web Sockets en HTML5

- En el lado del cliente, WebSocket está ya implementado en
  - Mozilla Firefox 8,
  - Google Chrome 4
  - Safari 5,
  - así como la versión móvil de Safari en el iOS 4.2.1
  - y en Internet Explorer 10.2

Web Sockets en HTML5

- Para establecer una conexión WebSocket, el cliente manda una petición de negociación WebSocket, y el servidor manda una respuesta de negociación WebSocket, como se puede ver en el siguiente ejemplo:
  - Petición del navegador al servidor:

```
GET /demo HTTP/1.1
Host: example.com
Connection: Upgrade
Sec-WebSocket-Key2: 12998 5 Y3 1 .P00
Sec-WebSocket-Protocol: sample
Upgrade: WebSocket
Sec-WebSocket-Key1: 4 @1 46546xW%0l 1 5
Origin: http://example.com

^n:ds[4U
```

Web Sockets en HTML5

- Para establecer una conexión WebSocket, el cliente manda una petición de negociación WebSocket, y el servidor manda una respuesta de negociación WebSocket, como se puede ver en el siguiente ejemplo:
  - Respuesta del servidor:

```
HTTP/1.1 101 WebSocket Protocol Handshake
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Origin: http://example.com
Sec-WebSocket-Location: ws://example.com/demo
Sec-WebSocket-Protocol: sample

8jKS'y:G*Co,Wxa-
```

Web Sockets en HTML5

- A la hora de implementar WebSockets, en la parte del cliente no hay mucha dificultad y tan solo con mirar la documentación podemos tenerlo realizado rapidamente pero la parte del servidor puede cambiar bastante dependiendo el lenguaje que usemos, etc.
- Entonces nace socket.io para facilitarnos la vida proporcionandonos más o menos el mismo código tanto en cliente como en servidor.

Socket.IO

- Es una librería open source con una amplia comunidad que nos ayudará a contruir aplicaciones con conexión persitente entre cliente y servidor. Por lo que contaremos con librerías para cada lado.
- Su web oficial es: http://socket.io/
- En un principio estaba pensado para trabajar con servidores escritos en NodeJS pero gracias a su éxito ha sido portado a muchos otros lenguajes como enseñaré en la siguiente lista:
  - Go: https://github.com/googollee/go-socket.io
  - Python: https://pypi.python.org/pypi/gevent-socketio/0.3.6
  - Java: https://github.com/mrniko/netty-socketio
  - y muchos más...

Socket.IO

- También la parte cliente ha sido portada a diferentes escenarios como puede ser:
  - Python-client: https://pypi.python.org/pypi/socketIO-client
  - Unity3D: https://github.com/NetEase/UnitySocketI
  - Java: https://github.com/nkzawa/socket.io-client.java
  - y muchos más como Android, iOS, etc.

Socket.IO

- Instalación
- O bien con npm install socket.io --save
- O bien crear el package.json del proyecto y con npm install

```
{

  "name": "chat-openwebinars",

  "version": "0.0.1",

  "description": "Aplicación de prueba en openwebinars",

  "dependencies": {

    "express": ">= 4.10.2",

    "socket.io": ">= 1.3.5"

  }

}
```

Socket.IO

Server

```
var numClients = 0;

io.on('connection', function(socket) {
    numClients++;
    io.emit('stats', { numClients: numClients });

    console.log('Connected clients:', numClients);

    socket.on('disconnect', function() {
        numClients--;
        io.emit('stats', { numClients: numClients });

        console.log('Connected clients:', numClients);
    });
});
```

Client

```
<script src="/socket.io/socket.io.js"></script>
<script>
    var socket = io.connect('/');
    socket.on('stats', function(data) {
        console.log('Connected clients:', data.numClients);
    });
</script>
```