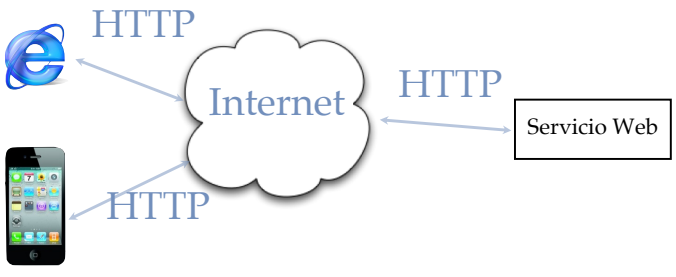
Desarrollo de servicios web con Node.js, Socket.io y MongoDB

Seminario práctico

Introducción

- Los servicios web ofrecen una funcionalidad a través de una url



Introducción

- La idea es mostraros como construir un servicio web utilizando las tecnologías Node.js, Socket.io y MongoDB

Introducción

- Tened en cuenta que para interactuar con los servicios web existen distintos protocolos que funcionan sobre HTTP.
- El estándar es SOAP (Simple Object Access Protocol), similar conceptualmente a RPC.
- Nosotros vamos a interactuar con los servicios de manera asíncrona (mediante eventos o notificaciones) y usando JSON+REST.

Introducción a JS

- Vamos a usar JavaScript para desarrollar servicios y clientes web para dichos servicios
- JavaScript es un lenguaje de programación para la web (HTML es el lenguaje de definición de vistas)
- Ahora una pequeña introducción...

Introducción a JS

- Tiene una sintaxis similar a c, c++ o java.
- Es interpretado y tiene tipado dinámico y débil
- Las funciones son un tipo especial de datos: se pueden pasar como parámetros o asociarse a variables
- La descripción de los objetos se hace en JSON (JavaScript Object Notation)

Ejemplo JSON

Es una colección de pares clave/valor

```
{
    "clave": "valor",
    "clave": "valor"
}
```

```
{
    "@id": "ssn:SensingDevice",
    "@type": "owl:Class",
    "rdfs:subClassOf": [  {"@id": "ssn:Device" },
     {"@id": "ssn:Sensor"}]
}
```

Se puede anidar

Introducción a JS

Variables var a = "hola"; a = 56;

Condicionales if...else... switch...

bucles for... while... do...while...

funciones

```
function suma(a,b){
    return a+b;
}
suma(3,5); //8
```

```
var f = function(a,b){
    return a+b;
};
f(3,5); //8
```

Introducción a JS

Objetos

```
var v = {a:1, b:2, c:"hola",
suma: function(a,b){
return a+b;
}};
```

```
v.c; //"hola"
v["c"]; //"hola"
v.suma(2,3); //5
```

Tipos de datos predefinidos:

Array

```
var v = [1,2,3];
v.push(5); //[1,2,3,5]
v.length; //4
v.splice(2,1); //[1,2,5]
```

Introducción a JS

Tipos de datos predefinidos:

String

```
var v = "hola";
v.charAt(0); //"h"
v.indexOf("o"); //1
v.split("l"); //["ho","a"]
v+"adios"; //"holaadios"
```

Date

```
var v = new Date(); //ahora
```

Math

```
Math.sqrt(4); //2
```

Además: Number, Boolean y RegExp

Node.js

- Node.js es una plataforma que permite desarrollar servicios web en JavaScript
- La programación sobre Node.js se hace de manera asíncrona:
  - Las funciones no devuelven nada y son no bloqueantes
  - Debemos pasar como parámetro a cualquier función un "callback"

Ejemplo: Hola Mundo

```
var http = require("http");
var httpServer = http.createServer(
  function(request, response) {
   console.log(request.headers);
   response.writeHead(200, {"Content-Type": "text/plain"});
   response.write("Hola mundo");
   response.end();
  }
);
httpServer.listen(8080);
console.log("Servicio HTTP iniciado");
```

Ejemplo: Hola Mundo

Instala Node.js
Guardalo en un fichero: holaMundo.js y ejecutalo con Node.js en el localhost:
$ node holaMundo.js
Abre un navegador y pon la url:
http://127.0.0.1:8080/ (localhost)

Ejemplo: Calculadora REST

```
var http = require("http");
var url = require("url");

function calcular(operacion, val1, val2) {
   if (operacion=="sumar") return val1+val2;
   else if (operacion == "restar") return val1-val2;
   else if (operacion == "producto") return val1*val2;
   else if (operacion == "dividir") return val1/val2;
   else return "Error: Par&aacute;metros no v&aacute;lidos";
}

var httpServer = http.createServer(
   function(request, response) {
    var uri = url.parse(request.url).pathname;
    var output = "";
    while (uri.indexOf('/') == 0) uri = uri.slice(1);
    var params = uri.split("/");
    if (params.length >= 3) {
        var val1 = parseFloat(params[1]);
        var val2 = parseFloat(params[2]);
        var result = calcular(params[0], val1, val2);
        output = result.toString();
    }
    else output = "Error: El n&uacute;mero de par&aacute;metros no es v&aacute;lido";

    response.writeHead(200, {"Content-Type": "text/html"});
    response.write(output);
    response.end();
   }
);
httpServer.listen(8080);
console.log("Servicio HTTP iniciado");
```

http://127.0.0.1:8080/sumar/3/4

Cliente de Calculadora

```
<html>
   <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Calculadora</title>
   </head>
   <body>
    <form action="javascript:void(0);" onsubmit="javascript:enviar();">
        Valor1: <input type="label" id="val1" /><br />
        Valor2: <input type="label" id="val2" /><br />
        Operaci&oacute;n:
        <select id="operacion">
            <option value="sumar">Sumar</option>
            <option value="restar">Restar</option>
            <option value="producto">Producto</option>
            <option value="dividir">Dividir</option>
        </select><br />
        <input type="submit" value="Calcular" />
    </form>
    <span id="resul"></span>
   </body>
   <script type="text/javascript">
    var serviceURL = " http://127.0.0.1:8080"; // servicio arrancado en localhost:8080
    function enviar() {
        var val1 = document.getElementById("val1").value;
        var val2 = document.getElementById("val2").value;
        var oper = document.getElementById("operacion").value;
        var url = serviceURL+"/"+oper+"/"+val1+"/"+val2;
        //  alert("He formado esta URL: " + url);
        var httpRequest = new XMLHttpRequest();
        httpRequest.onreadystatechange = function() {
        if (httpRequest.readyState === 4){
            var resultado = document.getElementById("resul");
            resultado.innerHTML = httpRequest.responseText;
           }
        };
        httpRequest.open("GET", url, true);
        httpRequest.send();
    }
   </script>
</html>
```

http://127.0.0.1:8080/sumar/3/4

Servidor con web cliente

- Cuando entramos a la web general http://127.0.0.1:8080 El propio servidor muestra una web con el interfaz de la calculadora.
- Ver documentación…

```
var httpServer = http.createServer(
  function(request, response) {
    var uri = url.parse(request.url).pathname;
    if (uri=="/") uri = "/calc.html";
    var fname = path.join(process.cwd(), uri);
    fs.exists(fname, function(exists) {
      if (exists) {
          fs.readFile(fname, function(err, data){
            if (!err) {
              var extension = path.extname(fname).split(".")[1];
              var mimeType = mimeTypes[extension];
              response.writeHead(200, mimeType);
              response.write(data);
              response.end();
            }
```

Socket.io

- Socket.io (módulo de node.js) permite enviar notificaciones a los clientes de un servicio
- Los clientes mantienen una conexión (un "WebSocket") con el servicio
- Cuando el servicio tiene datos nuevos, los notifica al cliente

Socket.io

- Publish-subscribe (eventos)
  - 'connect' – Conexión correcta
  - 'connecting' – Cliente está intentando conexión
  - 'disconnect' –
  - 'connect failed' -
  - 'error' - no puede ser tratado mediante cualquier otro evento por defecto.
  - 'message' – Informacion recibida a través de la función "send" que permite enviar información arbitrario (definida por el usuario)
    - La función emit es la standard que permite en Socket.io notificar eventos.
  - 'anything' - Se ha recibido un evento definido por el usuario (ninguno de los anteriores)
  - 'reconnect' / 'reconnecting' – Cliente trata de reconectarse a un servicio.
  - 'reconnect failed' -

Ejemplo: Clientes conectados

```
var http = require("http");
var url = require("url");
var fs = require("fs");
var path = require("path");
var socketio = require("socket.io");

var httpServer = http.createServer(
   function(request, response) {
   console.log("Peticion invalida: "+uri);
    response.writeHead(200, {"Content-Type": "text/plain"});
    response.write('404 Not Found\n');
    response.end();
   }
);
httpServer.listen(8080);
var io = socketio.listen(httpServer);

var allClients = new Array();
io.sockets.on('connection',
   function(client) {
    allClients.push(client.request.connection.remoteAddress + ":" +
client.request.connection.remotePort);
    io.sockets.emit('all-connections', allClients);
    client.on('output-evt', function (data) {
        client.emit('output-evt', 'Hola Cliente!');
    });
    client.on('disconnect', function() {
        var index =
allClients.indexOf(client.request.connection.remoteAddress + ":" +
client.request.connection.remotePort);
        if (index != -1) {
            allClients.splice(index, 1);
            io.sockets.emit('all-connections', allClients);
        }
        console.log('El usuario '+
client.request.connection.remoteAddress + ":" +
client.request.connection.remotePort +' se ha desconectado');
    });
   }
);

console.log("Servicio Socket.io iniciado");
```

Cliente del servicio

```
<html>
   <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Connections</title>
   </head>
   <body>
    <span id="mensaje_servicio"></span>
    <div id="lista_usuarios"></div>
   </body>
   <script src="/socket.io/socket.io.js"></script>
   <script type="text/javascript">
    function mostrar_mensaje(msg){
        var span_msg = document.getElementById('mensaje_servicio');
        span_msg.innerHTML = msg;
    }
    function actualizarLista(usuarios){
        var listContainer = document.getElementById('lista_usuarios');
        listContainer.innerHTML = '';
        var listElement = document.createElement('ul');
        listContainer.appendChild(listElement);
        var num = usuarios.length;
        for(var i=0; i<num; i++) {
                var listItem = document.createElement('li');
                listItem.innerHTML = usuarios[i].address+":"+usuarios[i].port;
                listElement.appendChild(listItem);
            }
         }
    var serviceURL = document.URL;
    var socket = io.connect(serviceURL);
    socket.on('connect', function(){
        socket.emit('output-evt', 'Hola Servicio!');
    });
    socket.on('output-evt', function(data) {
        mostrar_mensaje('Mensaje de servicio: '+data);
    });
    socket.on('all-connections', function(data) {
        actualizarLista(data);
    });
    socket.on('disconnect', function() {
        mostrar_mensaje('El servicio ha dejado de funcionar!!');
    });
   </script>
</html>
```

MongoDB

- MongoDB es una base de datos tipo NoSQL
- No hay tablas, hay colecciones de entradas JSON
- Cada entrada puede tener un conjunto de claves y valores arbitrario
- Podemos hacer "queries" tipo SQL
- Hay drivers de MongoDB para Node.js

MongoDB

Comandos básicos:

- Para acceder: $ mongo
- Para visualizar las bd: $ show dbs;
- Para acceder a una bd: $ use nombre_bd;
- Para visualizar las colecciones: $ show collections;
- Para visualizar los registros: $ db.nombre_colección.find();
- Para visualizar los registros con un formato más visual: $ db.nombre_colección.pretty();

Ejemplo: MongoDB + Socket.io

```
var http = require("http");
var url = require("url");
var socketio = require("socket.io");
var MongoClient = require('mongodb').MongoClient;
var MongoServer = require('mongodb').Server;

var httpServer = http.createServer(
   function(request, response) {
    console.log("Peticion invalida: "+uri);
    response.writeHead(200, {"Content-Type": "text/plain"});
    response.write('404 Not Found\n');
    response.end();
   }
);

var mongoClient = new MongoClient(new MongoServer('localhost', 27017));
mongoClient.connect("mongodb://localhost:27017/mibd", function(err, db) {
   httpServer.listen(8080);
   var io = socketio.listen(httpServer);

   db.createCollection("test", function(err, collection){
    io.sockets.on('connection',
    function(client) {
        client.emit('my-address', ´host:client.request.connection.remoteAddress,
port:client.request.connection.remotePort});
        client.on('poner', function (data) {
            collection.insert(data, {safe:true}, function(err, result) {});
        });
        client.on('obtener', function (data) {
            collection.find(data).toArray(function(err, results){
                client.emit('obtener', results);
            });
        });
    });
    });
});

console.log("Servicio MongoDB iniciado");
```

Cliente del servicio

```
<html>
   <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>MongoDB Test</title>
   </head>
   <body>
    <div id="resultados"></div>
   </body>
   <script src="/socket.io/socket.io.js"></script>
   <script type="text/javascript">
    function actualizarLista(usuarios){
        var listContainer = document.getElementById('resultados');
        listContainer.innerHTML = '';
        var listElement = document.createElement('ul');
        listContainer.appendChild(listElement);
        var num = usuarios.length;
        for(var i=0; i<num; i++) {
              var listItem = document.createElement('li');
              listItem.innerHTML = JSON.stringify(usuarios[i]);
              listElement.appendChild(listItem);
            }
        }

    var serviceURL = document.URL;
    var socket = io.connect(serviceURL);

    socket.on('my-address', function(data) {
        var d = new Date();
        socket.emit('poner', {host:data.host, port:data.port, time:d});
        socket.emit('obtener', {host: data.address});
    });
    socket.on('obtener', function(data) {
        actualizarLista(data);
    });
    socket.on('disconnect', function() {
        actualizarLista({});
    });
   </script>
</html>
```