monkkee

Thu, 6 Oct. 2016 12:46 PM

MatlabWebSocket

https://github.com/jebej/MatlabWebSocket

First of all install JSON Parser x Matlab

JSONlab

http://it.mathworks.com/matlabcentral/fileexchange/33381-jsonlab--a-toolbox-to-encode-decode-json-files

The most popular and mature Matlab implementation is JSONlab, which was started in 2011 by a Qianqian Fang, a medical researcher at Massachusetts General Hospital. It is available on the Matlab File Exchange, on SourceForge and via a Subversion repository. The latest version is 1.0beta that includes BSON, a variant of JSON used in MongoDB, which compresses JSON data into a binary format. JSONlab is based in part on earlier JSON-Matlab implementations that are now deprecated: JSON Parser (2009) by Joel Feenstra, another JSON Parser (2011) by François Glineur, and Highly portable JSON-input parser (2009) by Nedialko.

JSONlab converts both strings and files given by filename directly into Matlab structures and vice-versa. For example:

```
>> loadjson('{"this": "that", "foo": [1,2,3], "bar": ["a", "b", "c"]}')
ans =
    this: 'that'
    foo: [1 2 3]
    bar: {'a'  'b'  'c'}

>> s = struct('this', {'a', 'b'}, 'that', {1,2})
s =
1x2 struct array with fields:
    this
    that

>> j = savejson(s)
j =
{
    "s": [
        {
            "this": "a",
            "that": 1
        },
        {
            "this": "b",
            "that": 2
        }
    ]
}
```

JSONlab will nest structures as necessary and translates the JSON types into the appropriate Matlab types and vice-versa. JSONlab is well documented, easy and fast. It is reliable because it is wellmaintained, has been around for several years and has many users. It is open source and issues and contributions are welcomed.

Then install MatlabWebSocket

MatlabWebSocket is a simple library consisting of a websocket server and client for Matlab built on Java-WebSocket, a java implementation of the websocket protocol by Nathan Rajlich. It currently does not support encryption.

Installation

The required java library matlabwebsocket.jar located in /dist/ must be placed on the static java class path in Matlab. See the Matlab Documentation.

To add files to the static path, create a javaclasspath.txt file:

1. Create an ASCII text file and name the file javaclasspath.txt.
2. Enter the name of a Java class folder or jar file, one per line. For example:

   ```
   /Volumes/VSSD/workspace/Noself/WebSocketsExample/MatlabWebSocket-master/dist/matlabwebsocket.jar
   ```

   Save the file in your preferences folder. To view the location of the preferences folder, type:

   ```
   prefdir
   ```

   Alternatively, save the javaclasspath.txt file in your MATLAB startup folder: /Users/toni/Library/Application Support/MathWorks/MATLAB/R2016b

You must also add the webSocketServer.m and/or webSocketClient.m files located in /matlab/ file to the Matlab path or put them into workspace where you are developing your application.

Usage

The webSocketServer.m file is an abstract Matlab class. The behaviour of the server must therefore be defined by creating a subclass that implements the following methods:

```
    onOpen(obj,message,conn)
    onMessage(obj,message,conn)
    onError(obj,message,conn)
    onClose(obj,message,conn)
```

obj is the object instance of the subclass, it is implicitly passed by Matlab.

message is the message received by the server.

conn is a java object representing the client connection that cause the event. For example, if a message is received, the conn object will represent the client that sent the message.

See the echoServer.m file for an implementation example.

Example

Here is the example is an echo server modified, it only implements the 'onMessage' method and use JSONlab to parse incoming message.

```
classdef echoServer < matWebSocketServer
    %ECHOSERVER Summary of this class goes here
    %   Detailed explanation goes here

    properties
        last
    end

    methods
        function obj = echoServer(port)
            %Constructor
            obj=obj@matWebSocketServer(port);
        end

        function t = getLast(obj)
            t=obj.last;
        end
    end

    methods (Access = protected)
        function onMessage(obj,message,conn)
            % This function sends and echo back to the client
            obj.last=loadjson(message);
            obj.send(conn,message); % Echo
        end
    end
end
```

Run the echo server by making sure that the file is on the Matlab path and executing:

```
        i = echoServer(30000);
```

to start the server on port 30000.

To test the server, open the client.html in the /client/ folder in a modern web browser (really anything released after 2013). The port should already be set to 30000.

You can now connect and send messages. If the server is working properly, you will receive messages identical to the ones you send.

```
{"Name": "IoThingsWare", "Record": {"this": "that", "foo": [1,2,3], "bar": ["a", "b", "c"]}}
```

To see member of struct created by reading message coming from websocket

```
>> i.getLast

ans =

  struct with fields:

      Name: 'IoThingsWare'
    Record: [1×1 struct]

>> a=obj.last

>> a.Record.bar(2)

ans =

  cell

    'b'

>>
```

To close the server, go back to Matlab and type:

```
delete(i);
clear i;
```

Acknowledgments

This work was inspired by a websocket client matlab implementation matlab-websockets.

It relies on the Java-WebSocket library.

The html client was taken from the [cwebsocket]https://github.com/m8rge/cwebsocket repository.