TechOps-Websocket

Kevin Daum[1]
Application Developer
TechOps, Mount Pleasant, MI, USA
daum1kc@cmich.edu

Amninder S Narota[2]
Developer
TechOps, Mount Pleasant, MI, USA
narot1a@cmich.edu

12th December 2014

[1]Project Lead
[2]Project Manager/Developer

1 Websocket

This project is not intended to replace AJAX and is not strictly even a replacement for Comet/Long-poll (Although there are many cases where this makes sense).

1.1 Requirements

Before starting, make sure the following requirements/dependencies are fulfilled:

```
Twisted==14.0.0
autobahn==0.9.0
six==1.8.0
wsgiref==0.1.2
zope.interface==4.1.1
```

1.2 Commands

Following are the commands to execute web socket file:

1. Using Python:

   ```
   python ws.py
   ```

2. Using Twistd:

   ```
   twistd -y ws.py
   ```

Item 2 of Section 1.2 is the production level command which will generate two files "*twistd.log*" & "*twistd.pid*". "*twist.log*" will keep the log of the the running process and PId is the unique ID assigned to each process running on the process. One possible command to kill a process with PID 4235 would be:

```
kill -INT 4235
```

or

```
sudo kill -INT 4235
```

which ever suits the situation.

2 ws.py

There are two ports which the server listens to:

1. **8000**: When the Twisted Server (**Click here → **Twisted Network Programming Essentials[1]**) is running the server listens to port 8000 as defined in following code but following code is to set port number for web socket process:
2. **5000**: Websocket listens for change on port 5000.

```
import sys
from twisted.web.wsgi import WSGIResource
from twisted.internet import reactor, protocol
from twisted.web.server import Site
from twisted.python import log
from twisted.web.static import File
from twisted.application import service
from twisted.application.internet import TCPServer


from autobahn.twisted.websocket import WebSocketServerFactory, WebSocketServerProtoco
from broadcast.broadcast import BroadcastServerProtocol, BroadcastServerFactory


initialized = True
fixtures = ()



if len(sys.argv) > 1 and sys.argv[1] == 'debug':
```

```
    log.startLogging(sys.stdout)
    debug = True
else:
```

```
    debug = False
ServerFactory = BroadcastServerFactory
factory = ServerFactory("ws://localhost:5000",
                        debug = debug,
                        debugCodePaths = debug)
factory.protocol = BroadcastServerProtocol
factory.setProtocolOptions(allowHixie76 = True)
listenWS(factory)

webdir = File(".")
web = Site(webdir)

if __name__ == "__main__":
    reactor.listenTCP(8000, web)
    reactor.run()
else:
    application = service.Application("charserver")
    TCPServer(8000, web).setServiceParent(application)
```

3 broadcast.py

This file consists of two classes which is imported in *ws.py* as explained in **Section 2**

3.1 BroadcastServerProtocol(WebSocketServerProtocol)

This class is responsible to communicating network peers. **onMessage(self, payload, isBinary)** is the method which is listening to the port defined in **ws.py** of List Item 2 of Section 2 and broadcasts the message to the network peers.

```
import sys

from twisted.internet import reactor
from twisted.python import log
from twisted.web.server import Site
from twisted.web.static import File

from autobahn.twisted.websocket import WebSocketServerFactory,\
                                       WebSocketServerProtocol,\
                                       listenWS


class BroadcastServerProtocol(WebSocketServerProtocol):
    def onOpen(self):
        self.factory.register(self)
    def onMessage(self, payload, isBinary):
        if not isBinary:
            # msg = "{} from {}".format(payload.decode('utf8'), self.peer)
            msg = "{}".format(payload.decode('utf8'))
            self.factory.broadcast(msg)

    def connectionLost(self, reason):
        WebSocketServerProtocol.connectionLost(self, reason)
        self.factory.unregister(self)
```

4 index.html

Once the server is running and the message can be send through javascript websocket. Following is the sample html template for live chat over network:

4.1 Html code Snippet

```
<body>
   <h1>WebSocket Broadcast Demo</h1>
   <noscript>You must enable JavaScript</noscript>
   <form>
      <p>Broadcast Message:
         <input
           id="message"
           type="text"
           size="50"
           maxlength="50"
           value="Hello from Browser!"
         >
       </p>
   </form>
   <button onclick='broadcast();'>Broadcast Message</button>
   <pre
     id="log"
     style="height: 20em;
           overflow-y: scroll;
           background-color: #faa;"
   >
   </pre>
</body>
```

4.2 JS code Snippet

```
<head>
   <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.2.6/jquery.min.js"
     type="text/javascript" charset="utf-8"></script>
   <script type="text/javascript">
      var sock = null;
      var ellog = null;

      window.onload = function() {

         var wsuri;
         ellog = document.getElementById('log');

```

```
13              if (window.location.protocol === "file:") {
14                  wsuri = "ws://localhost:5000";
15                  // or the IP address of the machine ws running on
16              } else {
17                 wsuri = "ws://" + window.location.hostname + ":5000";
18              }
19
20              if ("WebSocket" in window) {
21                  sock = new WebSocket(wsuri);
22              } else if ("MozWebSocket" in window) {
23                 sock = new MozWebSocket(wsuri);
24              } else {
25                 log("Browser does not support WebSocket!");
26              }
27
28              if (sock) {
29                 sock.onopen = function() {
30                    log("Connected to " + wsuri);
31                 }
32
33                 sock.onclose = function(e) {
34                    log("Connection closed (wasClean = " +
35                                            e.wasClean +
36                                            ", code = " +
37                                            e.code +
38                                            ", reason = '" +
39                                            e.reason + "')");
40                    sock = null;
41                 }
42
43                 sock.onmessage = function(e) {
44                    log("Got echo: " + e.data);
45                 }
46              }
47          };
48
49          function broadcast() {
50                 var delay = 100;
51              var msg = document.getElementById('message').value;
52              if (sock) {
53                 sock.send(msg);
54                 log("Sent: " + msg);
55              } else {
56                 log("Not connected.");
57              }
```

```
58          };
59          function loop(sock, delay){
60              var dt = new Date();
61              var time = dt.getTime()/1000;
62              sock.send(time);
63              setTimeout(
64                      loop(sock, delay), /* Request next message */
65                      delay /* ..after 1 seconds */
66                  );
67          }
68
69           function log(m) {
70              ellog.innerHTML += m + '\n';
71              ellog.scrollTop = ellog.scrollHeight;
72           };
73       </script>
74   </head>
```

References

[1] Abe Fettig. *Twisted Network Programming Essentials.* O'Reilly Media, Inc., 2005.