package techbysample.websocket.sample1;

import java.io.IOException;

import java.util.Vector;

import javax.websocket.OnClose;

import javax.websocket.OnError;

import javax.websocket.OnMessage;

import javax.websocket.OnOpen;

import javax.websocket.server.ServerEndpoint;

import javax.websocket.Session;

/\*\*

\*

\* @author TechBySample.com

\*

\*/

@ServerEndpoint("/myechoendpoint")

public class EchoEndpoint {

Vector<Session> connessione=new Vector<Session>();

/\*\*

\*

\*Il metodo vine chiamato quando la connessione è stabilita.

\*

\* @param sessione

\*/

@OnOpen

public void onOpen(Session sessione) {

System.out.println(sessione.getId() + " has opened a connection");

try {

sessione.getBasicRemote().sendText("Connessione stabilita");

} catch (IOException ex) {

ex.printStackTrace();

}

}

/\*\*

\* Il metodo viene chiamato quando si chiude la connessione.

\*

\* Note: You cannot send messages to the client from this method

\*/

@OnClose

public void onClose(Session session) {

System.out.println("Session " + session.getId() + " has ended");

connessione.removeElement(session);

}

/\*\*

\* Il metodo è chiamato quando si riceve un messaggio a questo server endpoint. Il metodo

\* intercetta il messaggio e permette di agire di conseguenza.

\*/

@OnMessage

public void onMessage(String message, Session session) {

System.out.println("Message from " + session.getId() + ": " + message);

String username[] =new String[3] ;

username[0]="mancimichela";

username[1]="panfiliedoardo";

username[2]="melettimarco";

String password[]=new String[3];

password[0]="perioli";

password[1]="informatica";

password[2]="info";

try {

String x[] = message.split("#");

if(x[0].equals("R")){

String risposta="L#No";

for(int i=0;i<=username.length;i++) {

if(x[1].equals(username[i]) && x[2].equals(password[i])) {

risposta="L#Si";

}

}

session.getBasicRemote().sendText(risposta);

if(risposta.equals("L#Si")) {

connessione.add(session);

}

}else if(x[0].equals("M")) {

}

}

catch (NumberFormatException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

\* Il metodo viene chiamato quando c'è un errore.

\*

\* @param e

\*/

@OnError

public void onError(Throwable e) {

e.printStackTrace();

}

}