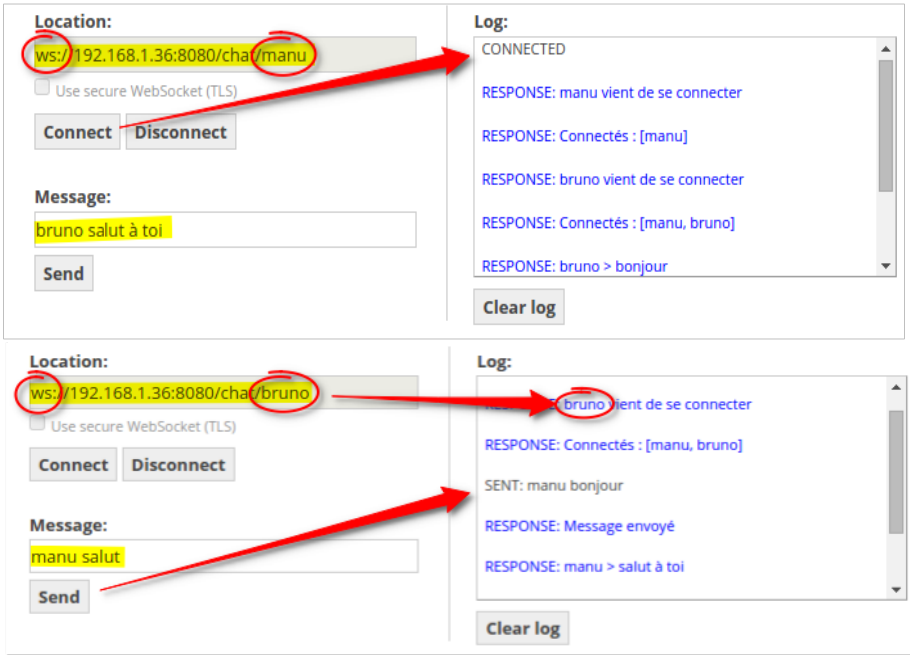
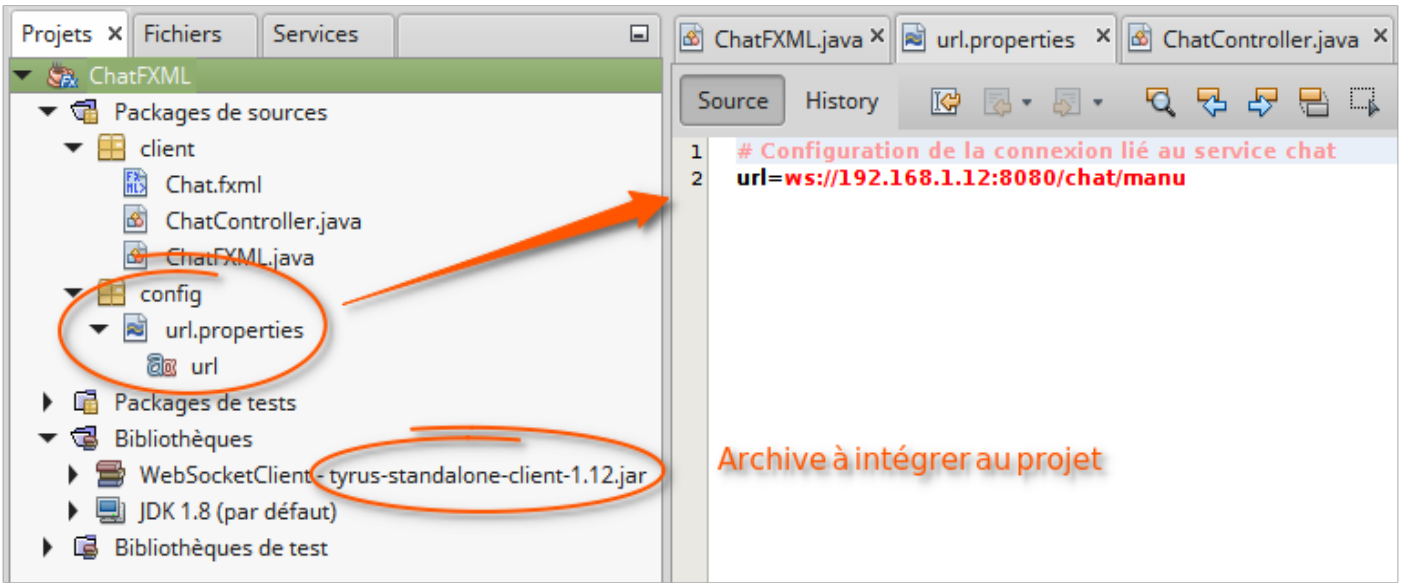
/home/manu/CloudStation/ProjetsJAVA/chat/src/java/service/ServiceChat.java

```
package service;

import java.io.*;
import java.util.*;
import javax.websocket.*;
import javax.websocket.server.*;

@ServerEndpoint("/{login}")
public class ServiceChat {
  private String login;
  private static HashMap<String, Session> clients = new HashMap<>();

   @OnOpen
   public void ouverture(Session client, @PathParam("login") String login) throws IOException {
     this.login = login;
     clients.put(login, client);
     envoyerATous(login + " vient de se connecter");
     envoyerATous("Connectés : "+clients.keySet().toString());
   }

   @OnClose
   public void fermeture() throws IOException {
     clients.remove(login);
     envoyerATous(login + " vient de se déconnecter");
     envoyerATous("Connectés : "+clients.keySet().toString());
   }

   @OnMessage
   public String chat(String réception) throws IOException{
     Scanner requête = new Scanner(réception);
     String destinataire = requête.next();
     String message = requête.nextLine();
     return envoyerMessage(destinataire, message) ? "Message envoyé" : "Cible inconnue";
   }

   private void envoyerATous(String message) throws IOException {
     for (Session client : clients.values())
       if (client.isOpen()) client.getBasicRemote().sendText(message);
   }

   private boolean envoyerMessage(String destinataire, String message) throws IOException {
     if (clients.containsKey(destinataire)) {
       clients.get(destinataire).getBasicRemote().sendText(login + " > " + message);
       return true;
     }
     return false;
   }
}
```



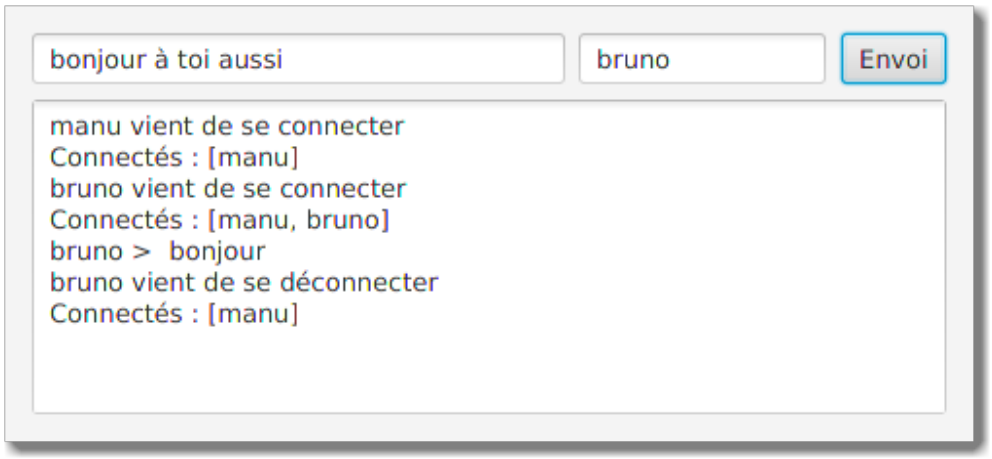
/home/manu/CloudStation/ProjetsJAVA/ChatWebSocket/src/client/ChatFXML.java

```
package client;

import javafx.application.Application;
import javafx.fxml.*;
import javafx.scene.*;
import javafx.stage.Stage;

public class ChatFXML extends Application {

  @Override
  public void start(Stage fenêtre) throws Exception {
    Parent root = FXMLLoader.load(getClass().getResource("Chat.fxml"));
    Scene scene = new Scene(root);
    fenêtre.setScene(scene);
    fenêtre.setTitle("Chat");
    fenêtre.show();
  }
}
```



/home/manu/CloudStation/ProjetsJAVA/ChatWebSocket/src/client/Chat.fxml

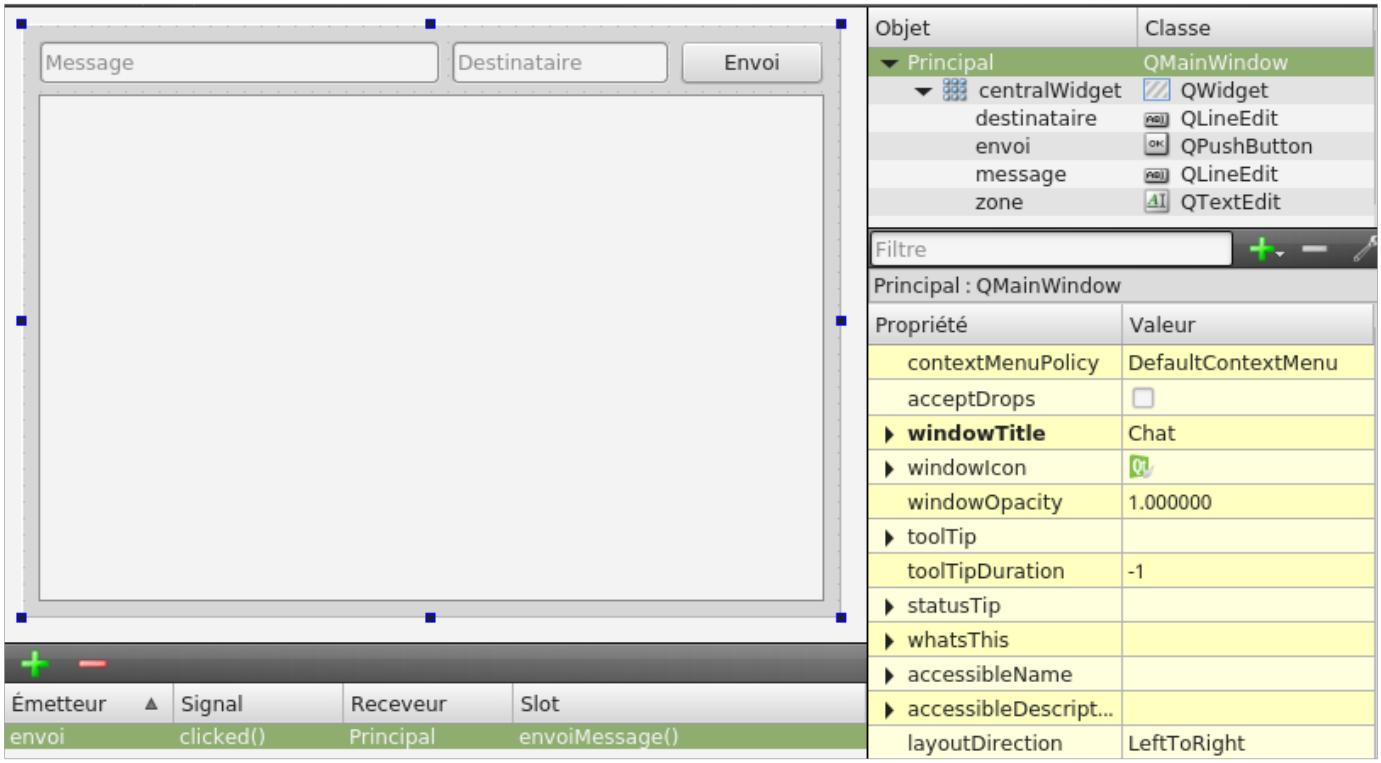
```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<AnchorPane prefHeight="380.0" prefWidth="536.0"
        xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/8" fx:controller="client.ChatController">
  <children>
    <Button layoutX="468.0" layoutY="14.0" onAction="#soumettre" text="Envoi"
        AnchorPane.rightAnchor="14.0" AnchorPane.topAnchor="14.0" />
    <TextField fx:id="message" layoutX="14.0" layoutY="14.0" prefHeight="25.0" prefWidth="312.0"
        promptText="Message" AnchorPane.leftAnchor="14.0"
        AnchorPane.rightAnchor="210.0" AnchorPane.topAnchor="14.0" />
    <TextField fx:id="destinataire" layoutX="333.0" layoutY="14.0" prefHeight="25.0" prefWidth="127.0"
        promptText="Destinataire" AnchorPane.rightAnchor="76.0" AnchorPane.topAnchor="14.0" />
    <TextArea fx:id="zone" layoutX="14.0" layoutY="48.0" prefHeight="322.0" prefWidth="512.0"
        AnchorPane.bottomAnchor="14.0" AnchorPane.leftAnchor="14.0"
        AnchorPane.rightAnchor="14.0" AnchorPane.topAnchor="48.0" />
  </children>
</AnchorPane>
```



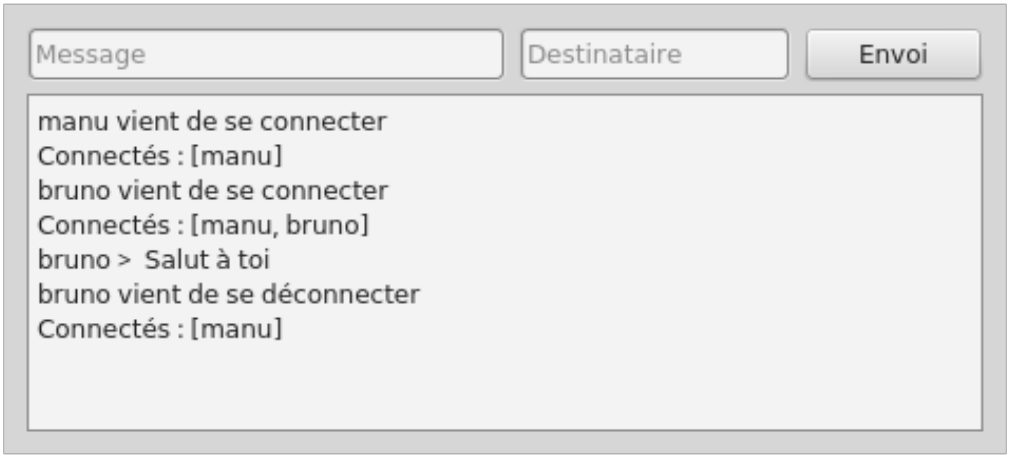
/home/manu/CloudStation/ProjetsJAVA/ChatWebSocket/src/client/ChatController.java

```
package client;

import java.io.*;
import java.net.*;
import java.util.*;
import javafx.fxml.*;
import javafx.scene.control.*;
import javax.websocket.*;

@ClientEndpoint
public class ChatController implements Initializable {
  @FXML private TextField message;
  @FXML private TextField destinataire;
  @FXML private TextArea zone;

  private String adresse;
  private Session session;

  @FXML
  private void soumettre() throws IOException {
    session.getBasicRemote().sendText(destinataire.getText()+' '+message.getText());
  }

  @Override
  public void initialize(URL url, ResourceBundle rb) {
    try {
      ResourceBundle config = ResourceBundle.getBundle("config/url");
      adresse = config.getString("url");
      WebSocketContainer conteneur = ContainerProvider.getWebSocketContainer();
      URI uri = URI.create(adresse);
      conteneur.connectToServer(this, uri);
    }
    catch (IOException | DeploymentException  ex) {
      Alert problème = new Alert(Alert.AlertType.WARNING);
      problème.setHeaderText("Impossible de se connecter");
      problème.showAndWait();
    }
  }

  @OnOpen
  public void ouverture(Session session) {
    this.session = session;
  }

  @OnMessage
  public void réception(String message) {
    zone.appendText(message+'\n');
  }
}
```

```
#ifndef PRINCIPAL_H
#define PRINCIPAL_H

#include "ui_principal.h"
#include <QWebSocket>

class Principal : public QMainWindow, private Ui::Principal
{
  Q_OBJECT

public:
  explicit Principal(QWidget *parent = 0);
private slots:
  void soumettre();
  void receptionMessage(QString texte);
private:
  void configuration();
private:
  QWebSocket service;
  QString url;
};

#endif // PRINCIPAL_H
```



```cpp
#include "principal.h"
#include <QSettings>

Principal::Principal(QWidget *parent) : QMainWindow(parent)
{
  setupUi(this);
  connect(&service, SIGNAL(textMessageReceived(QString)), this,
SLOT(receptionMessage(QString)));
  configuration();
  service.open(QUrl(url));
}

void Principal::configuration()
{
  QSettings config("url.ini", QSettings::IniFormat);
  url = config.value("serveur/url").toString();
}

void Principal::soumettre()
{
  QString requete = QString("%1 %2").arg(destinataire->text()).arg(message->text());
  service.sendTextMessage(requete);
}

void Principal::receptionMessage(QString texte)
{
  zone->append(texte);
}
```



```cpp
#include <QCoreApplication>
#include "servicechat.h"

int main(int argc, char *argv[])
{
  QCoreApplication application(argc, argv);
  ServiceChat chat;
  QObject::connect(&chat, SIGNAL(stop()), &application, SLOT(quit()));
  return application.exec();
}
```

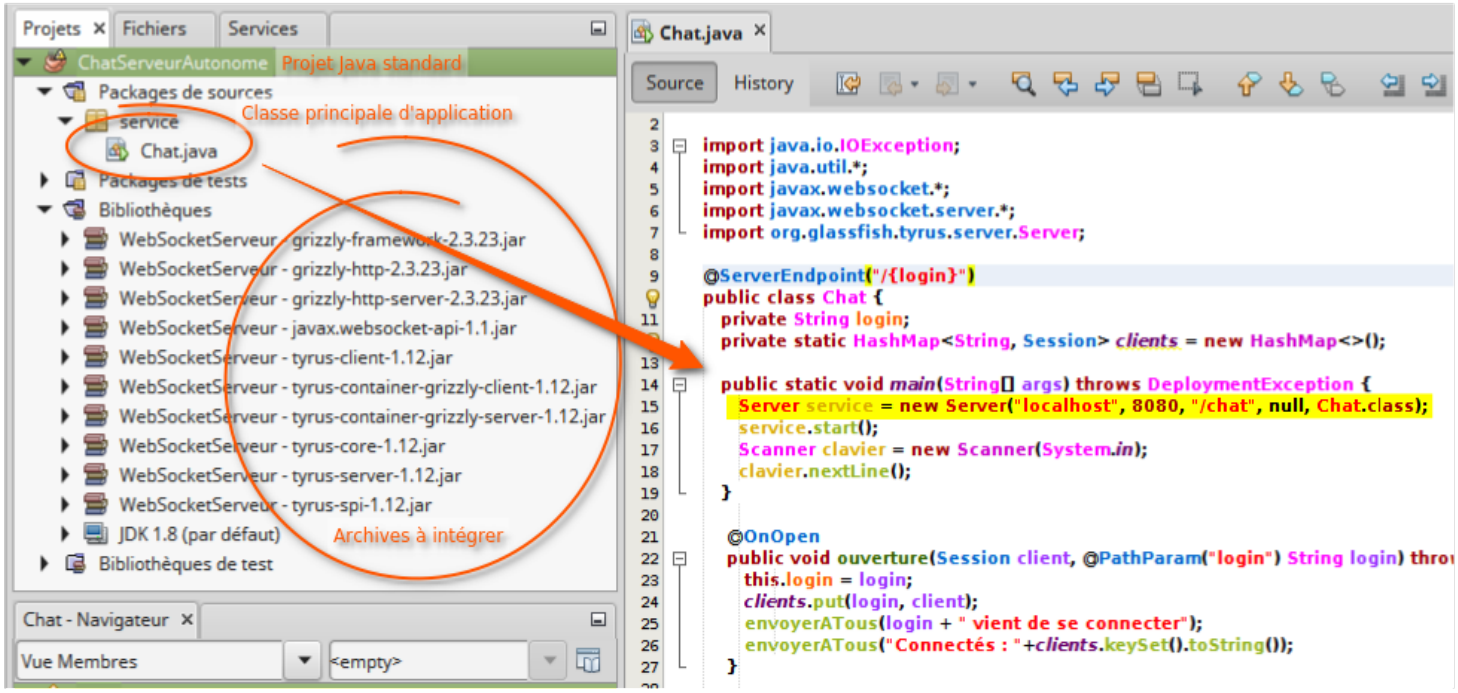
```
#ifndef SERVICECHAT_H
#define SERVICECHAT_H

#include <QObject>
#include <QWebSocketServer>
#include <QWebSocket>
#include <map>
using namespace std;

class ServiceChat : public QObject
{
  Q_OBJECT
public:
  explicit ServiceChat(QObject *parent = 0);
private:
  void envoyerATous(const QString &message);
  void listeDesConnectes();
signals:
  void stop();
public slots:
  void nouvelleConnexion();
  void receptionMessage(const QString &texte);
  void deconnexion();
private:
  QWebSocketServer service;
  map<QWebSocket *, QString> connectes;
};

#endif // SERVICECHAT_H
```

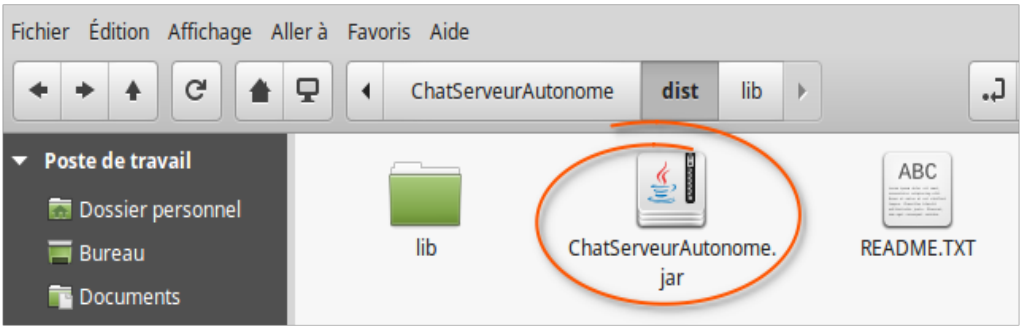
```cpp
#include "servicechat.h"

ServiceChat::ServiceChat(QObject *parent) : QObject(parent), service("chat",
QWebSocketServer::NonSecureMode, this)
{
  if (service.listen(QHostAddress::Any, 8080))
    connect(&service, SIGNAL(newConnection()), this, SLOT(nouvelleConnexion()));
  else stop();
}

void ServiceChat::nouvelleConnexion()
{
  QWebSocket *client = service.nextPendingConnection();
  QString nom = client->requestUrl().path();
  nom.remove('/');
  client->sendTextMessage("Bonjour "+nom);
  envoyerATous(nom+" vient de se connecter");
  connectes[client] = nom;
  listeDesConnectes();
  connect(client, SIGNAL(textMessageReceived(QString)), this,
SLOT(receptionMessage(QString)));
  connect(client, SIGNAL(disconnected()), this, SLOT(deconnexion()));
}

void ServiceChat::deconnexion()
{
  QWebSocket *client = (QWebSocket *) sender();
  client->deleteLater();
  QString nom = connectes[client];
  connectes.erase(client);
  envoyerATous(nom + " vient de se déconnecter");
  listeDesConnectes();
  disconnect(client, SIGNAL(textMessageReceived(QString)), this,
SLOT(receptionMessage(QString)));
  disconnect(client, SIGNAL(disconnected()), this, SLOT(deconnexion()));
}

void ServiceChat::receptionMessage(const QString &texte)
{
  if (texte=="stop") { stop(); return; }
  QWebSocket *client = (QWebSocket *) sender();
  QString expediteur = connectes[client];
  QString destinataire = texte.split(' ')[0];
  QString message = texte.split(destinataire)[1];
  bool trouve = false;
  for (auto &cible : connectes)
    if (cible.second == destinataire) {
      cible.first->sendTextMessage(expediteur+" > "+message);
      client->sendTextMessage("Message envoyé");
      trouve = true; break;
    }
  if (!trouve) client->sendTextMessage("Cible inconnue");
}

void ServiceChat::envoyerATous(const QString &message)
{
  for (auto &client : connectes) client.first->sendTextMessage(message);
}

void ServiceChat::listeDesConnectes()
{
  QString liste = "[";
  for (auto &client : connectes) liste.append(" "+client.second);
  liste.append(" ]");
  envoyerATous("Connectés : "+liste);
}
```

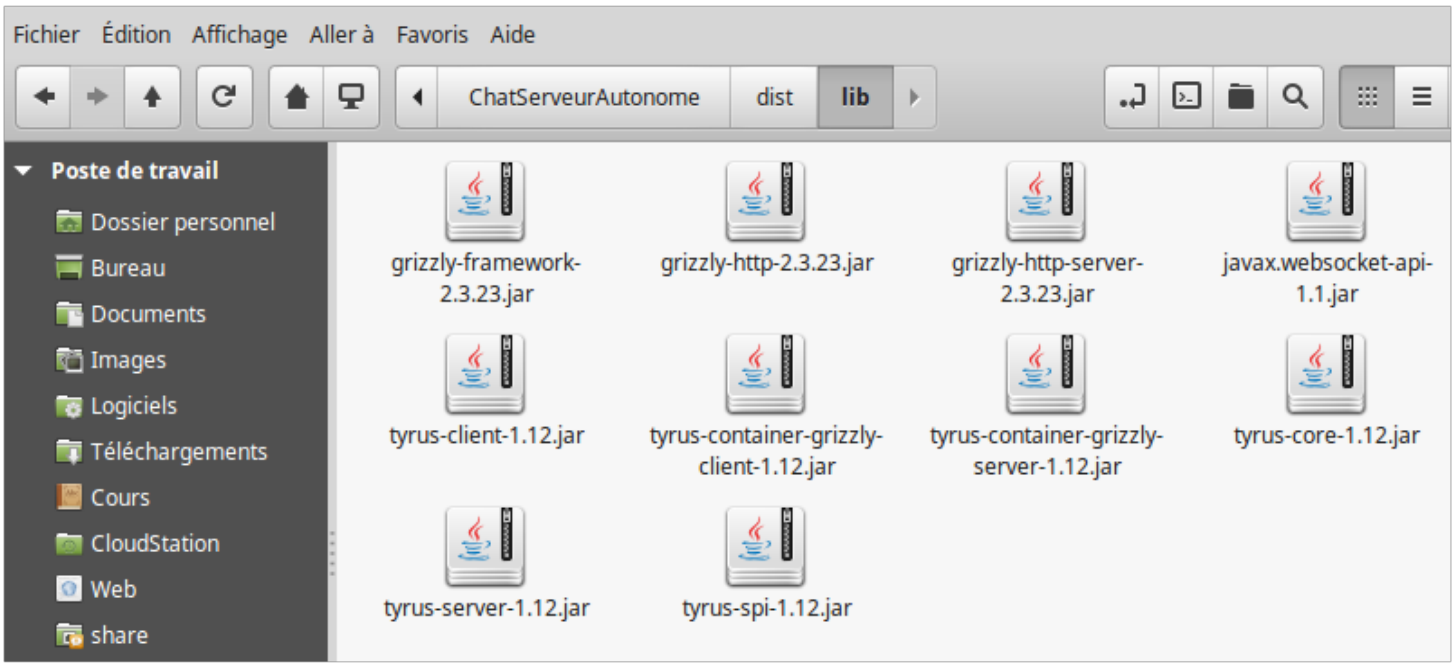
/home/manu/CloudStation/ProjetsJAVA/ChatServeurAutonome/src/service/Chat.java

```
package service;

import java.io.IOException;
import java.util.*;
import javax.websocket.*;
import javax.websocket.server.*;
import org.glassfish.tyrus.server.Server;

@ServerEndpoint("/{login}")
public class Chat {
  private String login;
  private static HashMap<String, Session> clients = new HashMap<>();

  public static void main(String[] args) throws DeploymentException {
    Server service = new Server("localhost", 8080, "/chat", null, Chat.class);
    service.start();
    Scanner clavier = new Scanner(System.in);
    clavier.nextLine();
  }

  @OnOpen
  public void ouverture(Session client, @PathParam("login") String login) throws IOException {
    this.login = login;
    clients.put(login, client);
    envoyerATous(login + " vient de se connecter");
    envoyerATous("Connectés : "+clients.keySet().toString());
  }

  @OnClose
  public void fermeture() throws IOException {
    clients.remove(login);
    envoyerATous(login + " vient de se déconnecter");
    envoyerATous("Connectés : "+clients.keySet().toString());
  }

  @OnMessage
  public String chat(String réception) throws IOException{
    Scanner requête = new Scanner(réception);
    String destinataire = requête.next();
    String message = requête.nextLine();
    return envoyerMessage(destinataire, message) ? "Message envoyé" : "Cible inconnue";
  }

  private void envoyerATous(String message) throws IOException {
    for (Session client : clients.values())
      if (client.isOpen()) client.getBasicRemote().sendText(message);
  }

  private boolean envoyerMessage(String destinataire, String message) throws IOException {
    if (clients.containsKey(destinataire)) {
      clients.get(destinataire).getBasicRemote().sendText(login + " > " + message);
      return true;
    }
    return false;
  }
}
```

```
manu@N550JV ~/CloudStation/ProjetsJAVA/ChatServeurAutonome/dist $ java -jar Chat
ServeurAutonome.jar
déc. 09, 2015 10:30:29 AM org.glassfish.grizzly.http.server.NetworkListener star
t
INFOS: Started listener bound to [0.0.0.0:8080]
déc. 09, 2015 10:30:29 AM org.glassfish.grizzly.http.server.HttpServer start
INFOS: [HttpServer] Started.
déc. 09, 2015 10:30:29 AM org.glassfish.tyrus.server.Server start
INFOS: WebSocket Registered apps: URLs all start with ws://localhost:8080
déc. 09, 2015 10:30:29 AM org.glassfish.tyrus.server.Server start
INFOS: WebSocket server started.
```


Projets
Fichiers
Services
ChatServeurAutonome
Projet Java standard
Packages de sources
service
Classe principale d'application
Chat.java
Packages de tests
Bibliothèques
WebSocketServeur - grizzly-framework-2.3.23.jar
WebSocketServeur - grizzly-http-2.3.23.jar
WebSocketServeur - grizzly-http-server-2.3.23.jar
WebSocketServeur - javax.websocket-api-1.1.jar
WebSocketServeur - tyrus-client-1.12.jar
WebSocketServeur - tyrus-container-grizzly-client-1.12.jar
WebSocketServeur - tyrus-container-grizzly-server-1.12.jar
WebSocketServeur - tyrus-core-1.12.jar
WebSocketServeur - tyrus-server-1.12.jar
WebSocketServeur - tyrus-spi-1.12.jar
JDK 1.8 (par défaut)
Archives à intégrer
Bibliothèques de test
Chat - Navigateur
Vue Membres
<empty>
Chat.java
Source
History
import java.io.IOException;
import java.util.*;
import javax.websocket.*;
import javax.websocket.server.*;
import org.glassfish.tyrus.server.Server;
@ServerEndpoint("/{login}")
public class Chat {
private String login;
private static HashMap<String, Session> clients = new HashMap<>();
public static void main(String[] args) throws DeploymentException {
Server service = new Server("localhost", 8080, "/chat", null, Chat.class);
service.start();
Scanner clavier = new Scanner(System.in);
clavier.nextLine();
}
@OnOpen
public void ouverture(Session client, @PathParam("login") String login) throu
this.login = login;
clients.put(login, client);
envoyerATous(login + " vient de se connecter");
envoyerATous("Connectés : "+clients.keySet().toString());
}


Fichier
Édition
Affichage
Aller à
Favoris
Aide
ChatServeurAutonome
dist
lib
Poste de travail
Dossier personnel
Bureau
Documents
lib
ChatServeurAutonome.jar
README.TXT


Fichier
Édition
Affichage
Aller à
Favoris
Aide
ChatServeurAutonome
dist
lib
Poste de travail
Dossier personnel
Bureau
Documents
Images
Logiciels
Téléchargements
Cours
CloudStation
Web
share
grizzly-framework-2.3.23.jar
grizzly-http-2.3.23.jar
grizzly-http-server-2.3.23.jar
javax.websocket-api-1.1.jar
tyrus-client-1.12.jar
tyrus-container-grizzly-client-1.12.jar
tyrus-container-grizzly-server-1.12.jar
tyrus-core-1.12.jar
tyrus-server-1.12.jar
tyrus-spi-1.12.jar