Example of WebSocket/STOMP using Java

# Simple WebSocket Server

We will use the WebSocket API to create a simple Server.

### Configuration: pom.xml

The pom would be the same as a typical JEE pom. Just put the following dependency:

<dependency>

<groupId>javax.websocket</groupId>

<artifactId>javax.websocket-api</artifactId>

<version>1.1</version>

<scope>provided</scope>

</dependency>

### Server Code

@ServerEndpoint(value = "/hello-stomp")

**public** **class** SimpleServerEndpoint {

**private** **static** **final** Logger ***LOGGER*** = LoggerFactory.*getLogger*(SimpleServerEndpoint.**class**);

**private** **static** **final** String ***DATE\_FORMAT*** = "dd-MM-yyyy HH:mm:ss";

@OnMessage

**public** String onMessage(String message, Session session) {

***LOGGER***.info("Message from client: `{}`", message);

String serverMessage = **new** StringBuilder(100).append("Message processed by SimpleServer at [").append(**new** SimpleDateFormat(***DATE\_FORMAT***).format(**new** Date())).append("]").append(message)

.toString();

**return** serverMessage;

}

}

# Simple WebSocket Java Client

### Configuration: pom.xml

The pom would be a simple one, just add these following dependencies:

<dependency>

<groupId>javax.websocket</groupId>

<artifactId>javax.websocket-api</artifactId>

<version>1.1</version>

</dependency>

<dependency>

<groupId>org.glassfish.tyrus.bundles</groupId>

<artifactId>tyrus-standalone-client</artifactId>

<version>1.12</version>

</dependency>

### Client Code

Define an Endpoint:

**public** **class** SimpleClientEndpoint **extends** Endpoint {

**private** **static** **final** Logger ***LOGGER*** = LoggerFactory.*getLogger*(SimpleClientEndpoint.**class**);

@Override

**public** **void** onOpen(Session session, EndpointConfig config) {

***LOGGER***.info("session opened");

session.addMessageHandler(String.**class**, **new** Whole<String>() {

@Override

**public** **void** onMessage(String text) {

***LOGGER***.info("recieved message from server: `{}`", text);

}

});

String message = "Hello from client";

***LOGGER***.info("trying to send message `{}` to server...", message);

**try** {

session.getBasicRemote().sendText(message);

***LOGGER***.info("message sent successfully");

} **catch** (IOException e) {

***LOGGER***.error("error sending message to server", e);

}

}

}

Then invoke this:

**public** **class** SimpleClient {

**public** **static** **void** main(String[] args) **throws** InterruptedException,

URISyntaxException, DeploymentException, IOException {

ClientEndpointConfig cec = ClientEndpointConfig.Builder.*create*()

.build();

ClientManager client = ClientManager.*createClient*();

client.connectToServer(SimpleClientEndpoint.**class**, cec, **new** URI(

"ws://localhost:8080/stomp-server/hello-stomp"));

Thread.*sleep*(10\_000);

}

}

Note that there is a 2-way communication from server/client. This is handled in the MessageHandler defined in SimpleClientEndpoint:

session.addMessageHandler(String.**class**, **new** Whole<String>() {

@Override

**public** **void** onMessage(String text) {

***LOGGER***.info("recieved message from server: `{}`", text);

}

});

<http://docs.spring.io/spring/docs/current/spring-framework-reference/html/websocket.html>

<https://tyrus.java.net/documentation/1.12/index/getting-started.html>

<https://www.websocket.org/echo.html>