TUTORIAL WEBSOCKET

1. Requisitos:

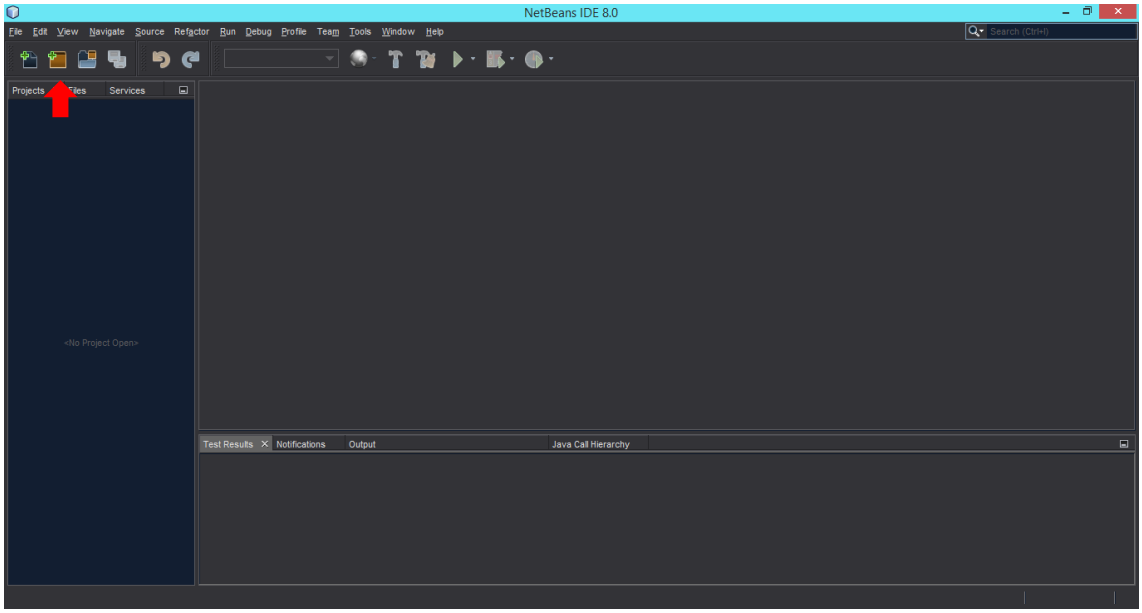
- ✓ Java 7 JDK (ou superior);
- ✓ Netbeans 8.0 (com Java EE);
- ✓ Maven (ou superior já incluído nativamente no Netbeans);
- ✓ GlassFish Server 4 Open Source Edition.

2. Implementação

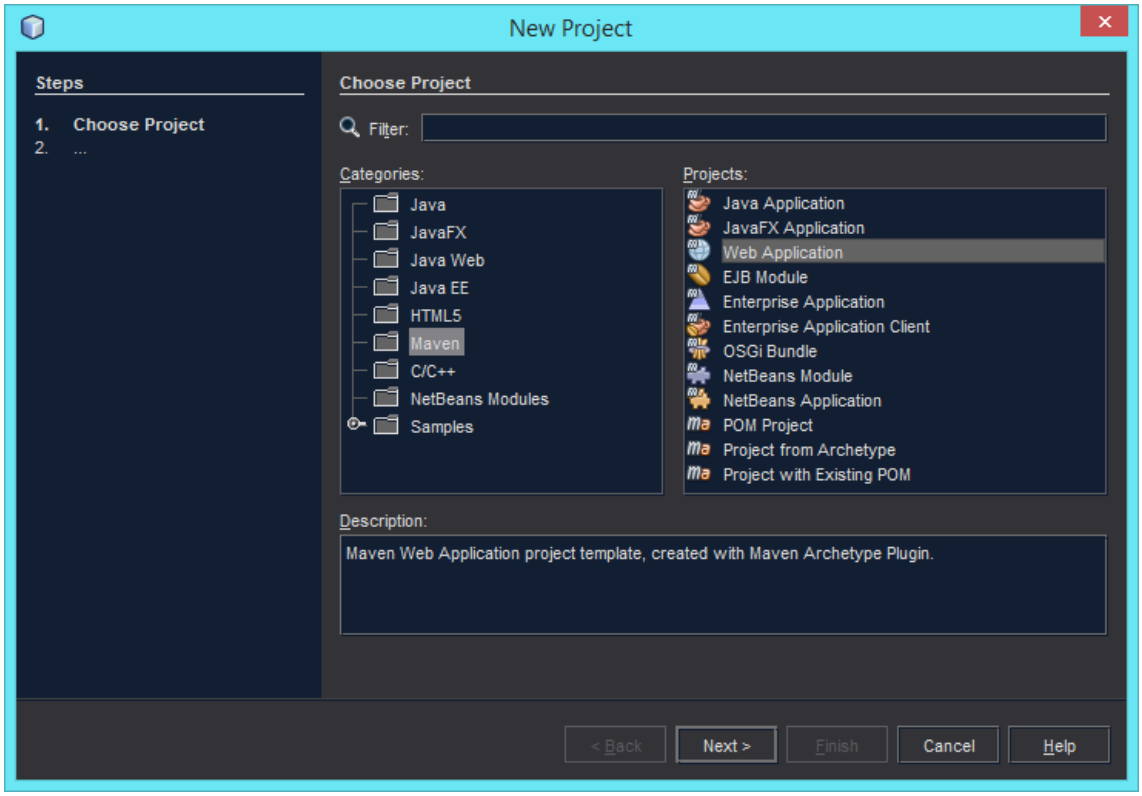
2.1 WebSocket Server

Etapa 1 – Criação do projeto

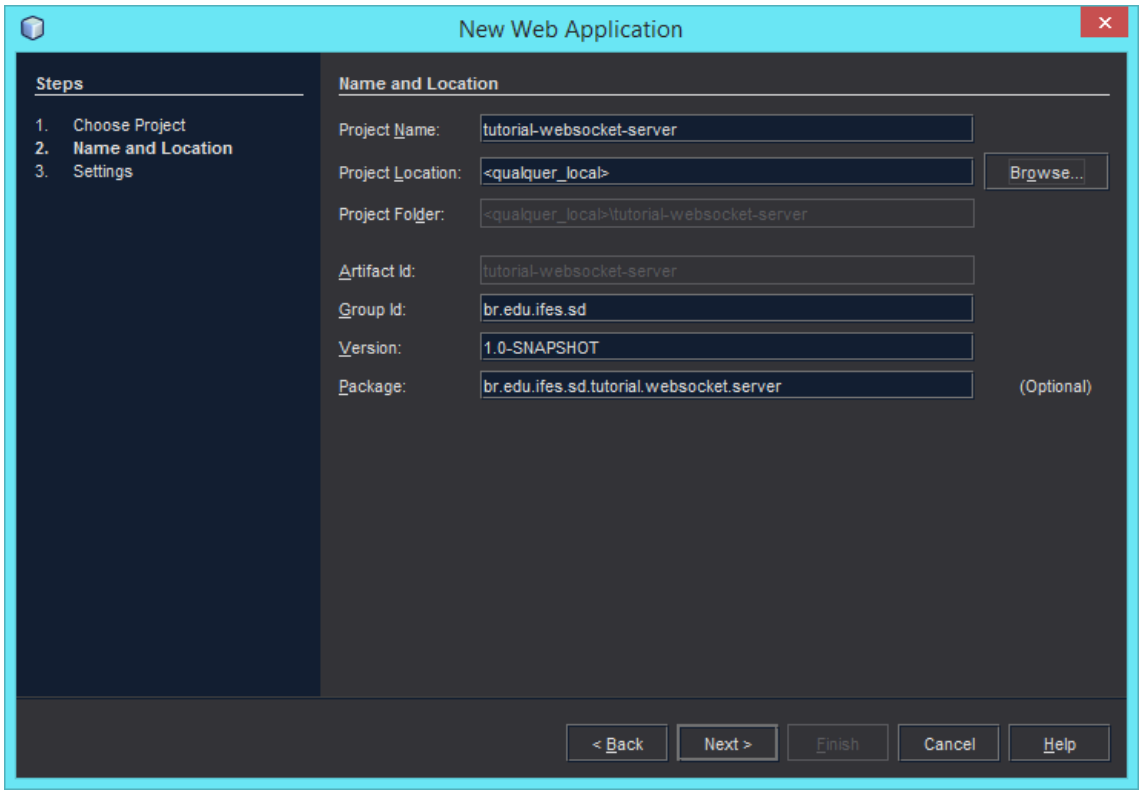
Nesta etapa, vamos criar um projeto **Maven** do tipo **WebApplication**. Clique em **New Project**.



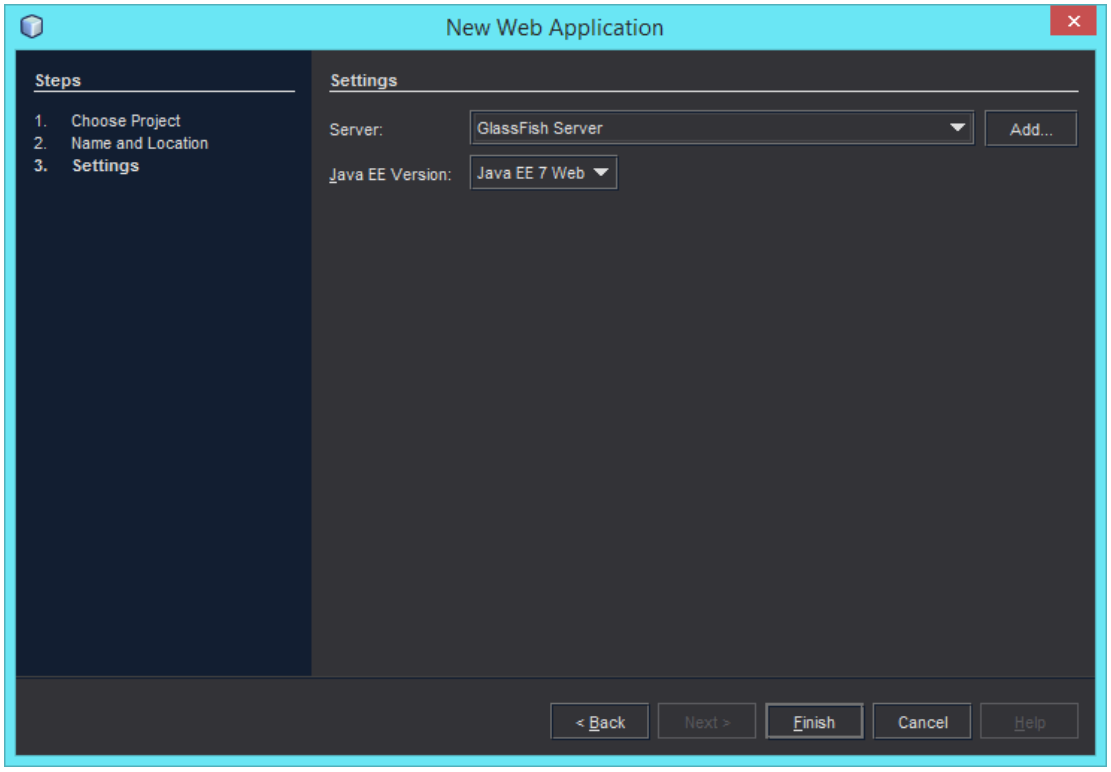
Em seguida, selecione a categoria **Maven** e escolha **Web Application**. Clique em **Next**.



Preencha os dados conforme a tabela abaixo, e selecione um local qualquer para o projeto. Clique em **Next**.



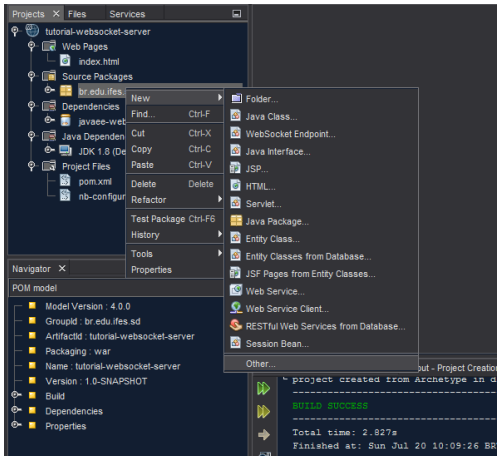
Agora, selecione o GlassFish como servidor da aplicação. (Caso você ainda não tenha o GlassFish 4.0 instalado em seu computador, faça-o primeiro).



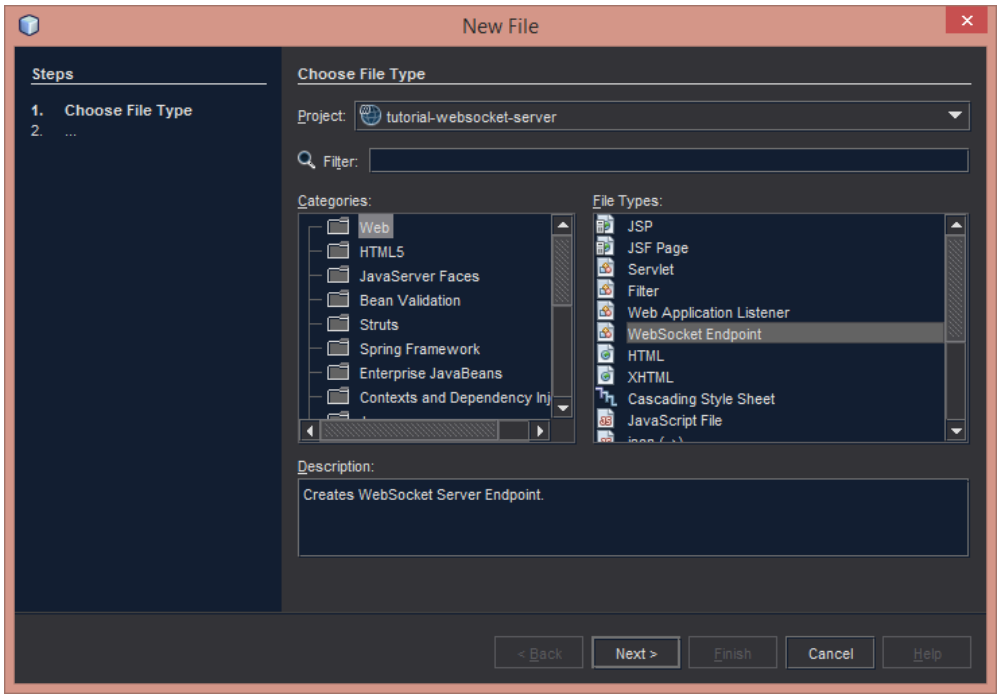
Etapa 2 – WebSocket Endpoint

Nesta etapa, vamos criar o WebSocket Endpoint. Segundo a documentação disponível no site da Oracle, um WebSocket Endpoint representa um objeto capaz de lidar com comunicações via websocket (tanto no lado do servidor, quanto no lado do cliente).

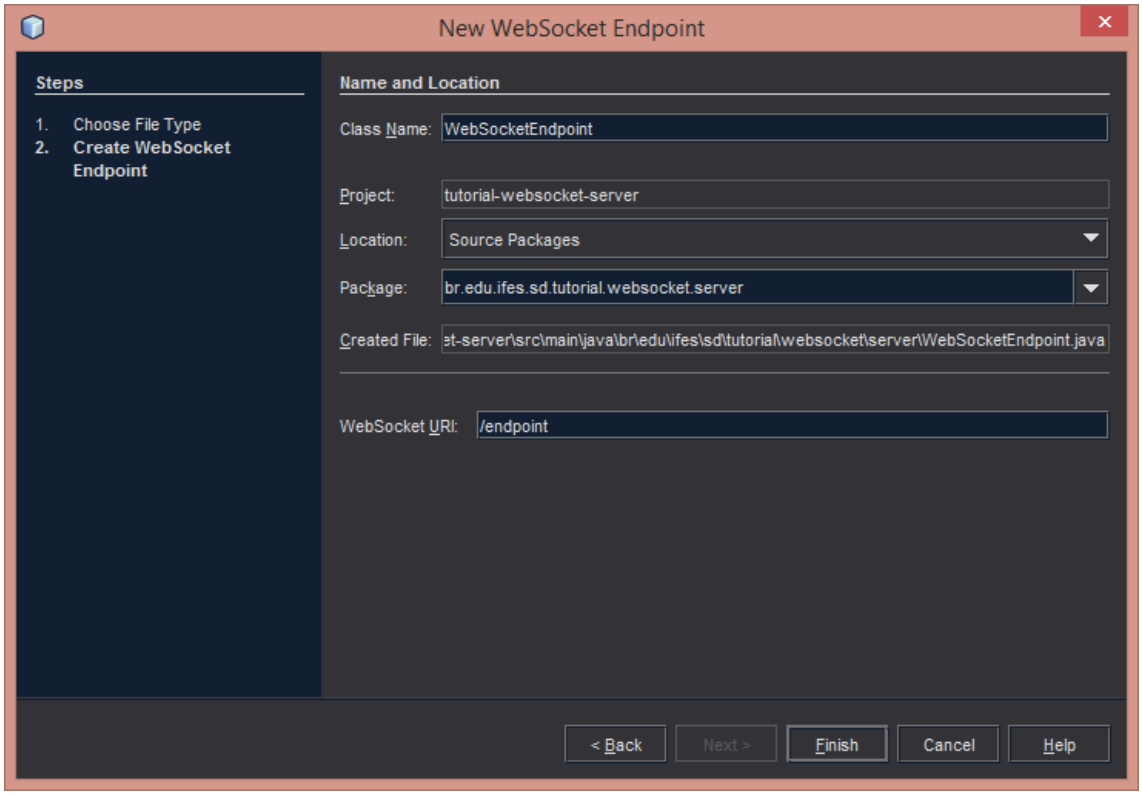
Vá para o pacote **br.edu.ifes.sd.tutorial.websocket.server**, clique com o botão direito sobre ele, vá em **New > Other**.



Escolha a categoria **Web** e selecione **WebSocket Endpoint**. Clique em **Next**.



Novamente, preencha os dados conforme a imagem abaixo. E cliquem em **Finish**.

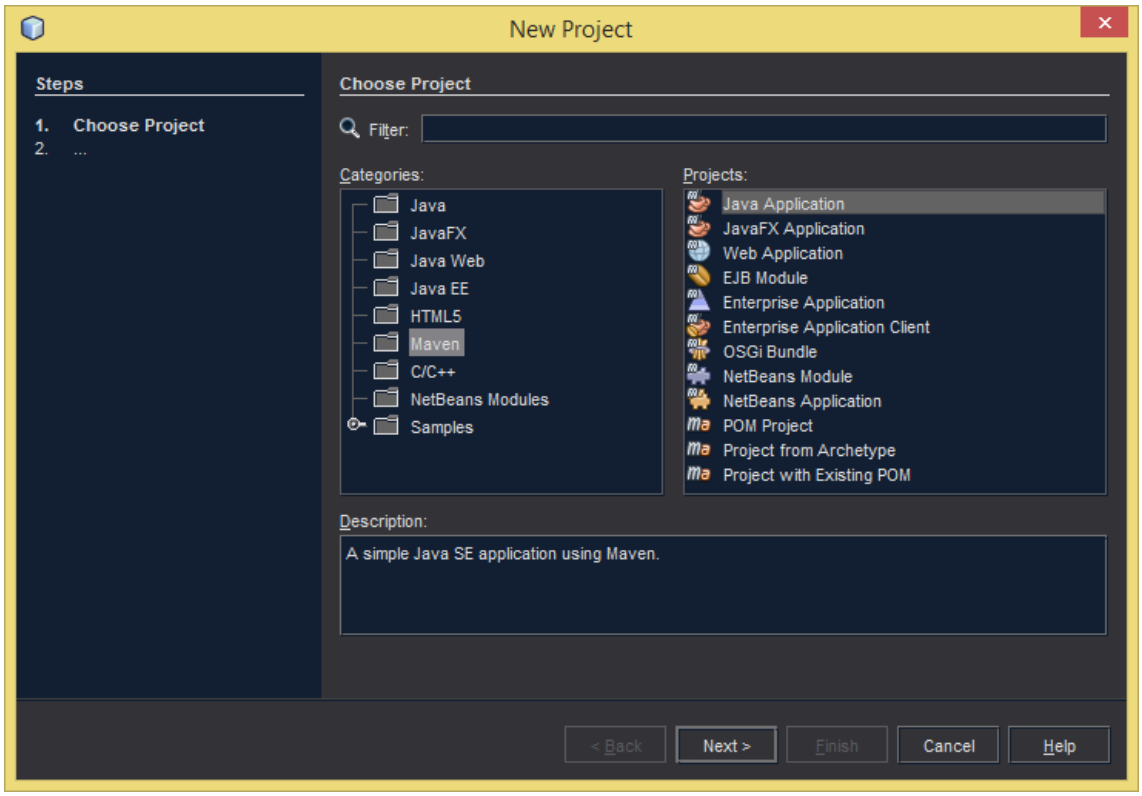


Agora, copie o conteúdo desta classe que está no **Anexo I**. Compile o projeto, caso dê algum erro, volte no tutorial e confira se está tudo certo. **Não execute o projeto ainda!**

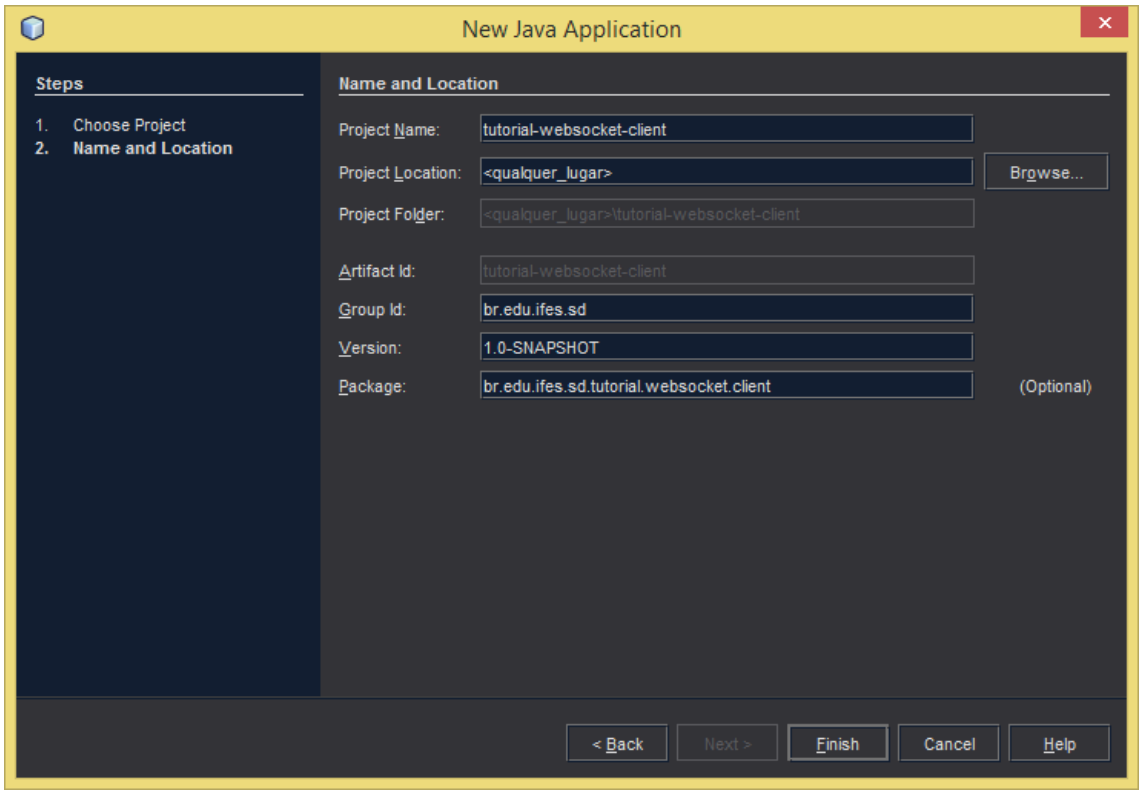
2.2 WebSocket Client

Etapa 1 – Criação do projeto

Nesta etapa, vamos criar um projeto Maven do tipo Java Application. Clique em **New Project**. E em seguida, escolha a categoria **Maven** e selecione **Java Application**. Clique em **Next**.



Preencha os dados conforme a figura abaixo. Clique em **Finish**.



Etapa 2 – Importando as dependências do projeto

Abra o arquivo **pom.xml**, e adicione o repositório e as bibliotecas necessárias.

```
<repositories>
    <repository>
        <id>java.net-promoted</id>
        <url>https://maven.java.net/content/groups/promoted/</url>
    </repository>
</repositories>

<dependencies>
    <dependency>
        <groupId>org.glassfish.tyrus</groupId>
        <artifactId>tyrus-client</artifactId>
        <version>1.0-rc3</version>
    </dependency>
    <dependency>
        <groupId>org.glassfish.tyrus</groupId>
        <artifactId>tyrus-container-grizzly</artifactId>
        <version>1.0-rc3</version>
    </dependency>
    <dependency>
        <groupId>javax</groupId>
        <artifactId>javaee-api</artifactId>
        <version>7.0-b80</version>
    </dependency>
</dependencies>
```

Etapa 3 – Criando WebSocket Endpoint

Novamente, vamos criar um WebSocket Endpoint só que agora para o cliente. Diferente do projeto Web, criaremos uma classe Java normal com o nome **ClientDesktopEndpoint.java**.

Depois de criada, copie o conteúdo desta classe que está no **Anexo II**.

Em seguida crie uma classe chamada ChatApp, e copie o conteúdo desta classe que está no **Anexo III**.

3. Execução

Nesta parte do tutorial, vá até o projeto **WebSocket Server**, compile e execute-o.

Agora que o servidor do WebSocket está executando. Vá ao projeto **WebSocket Client**, compile e execute duas ou mais instancias. O que aconteceu? Todos os clientes estão recebendo as mensagens?

Volte nos códigos que executamos, e veja como as coisas estão funcionando. Acredito que você conseguirá entender – pelo menos, por alto – o que está acontecendo. Tente rodar esta aplicação de forma distribuída, e veja se o resultado é o mesmo.

ANEXO I – WebSocketEndpoint.java

```java
package br.edu.ifes.sd.tutorial.websocket.server;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.websocket.EncodeException;
import javax.websocket.OnClose;
import javax.websocket.OnMessage;
import javax.websocket.OnOpen;
import javax.websocket.Session;
import javax.websocket.server.PathParam;
import javax.websocket.server.ServerEndpoint;

/**
 *
 * @author Lucas
 */
@ServerEndpoint("/websocket/{client-id}")
public class WebSocketEndpoint {

    private static final List<Session> clients = new ArrayList<>();

    @OnOpen
    public void onOpen(Session session) {
        clients.add(session);
    }

    @OnMessage
    public void onMessage(String message, @PathParam("client-id") String clientId) {
        try {
            for (Session client : clients) {
                client.getBasicRemote().sendObject(clientId + ": " + message);
            }
        } catch (IOException | EncodeException ex) {
            Logger.getLogger(WebSocketEndpoint.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }

    @OnClose
    public void onClose(Session peer) {
        clients.remove(peer);
    }

}
```

ANEXO II – ClientDesktopEndpoint.java

```java
package br.edu.ifes.sd.tutorial.websocket.client;

import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.websocket.ClientEndpoint;
import javax.websocket.OnError;
import javax.websocket.OnMessage;
import javax.websocket.OnOpen;
import javax.websocket.Session;

/**
 *
 * @author Lucas
 */
@ClientEndpoint
public class ClientDesktopEndpoint {

    @OnOpen
    public void onOpen(Session session) {
        try {
            System.out.println("Connected to endpoint: " + session.getBasicRemote());
            session.getBasicRemote().sendText("Hello");
        } catch (IOException ex) {
            Logger.getLogger(ClientDesktopEndpoint.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }

    @OnMessage
    public void onMessage(String message) {
        System.out.println(message);
    }

    @OnError
    public void onError(Throwable t) {
        Logger.getLogger(ClientDesktopEndpoint.class.getName()).log(Level.SEVERE,
null, t);
    }
}
```

ANEXO III – ClientApp.java

```
package br.edu.ifes.sd.tutorial.websocket.client;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.websocket.ContainerProvider;
import javax.websocket.DeploymentException;
import javax.websocket.Session;
import javax.websocket.WebSocketContainer;

/**
 *
 * @author Lucas
 */
public class ClientApp {

    public Session session;

    protected void start() {

        try {
            WebSocketContainer webSocketContainer = ContainerProvider.getWebSocketContainer();

            String uri = "ws://localhost:8080/tutorial-websocket-server/websocket/desktop-
client";

            System.out.println("Connecting to " + uri);

            session = webSocketContainer.connectToServer(ClientDesktopEndpoint.class,
URI.create(uri));
        } catch (DeploymentException | IOException ex) {
            Logger.getLogger(ClientApp.class.getName()).log(Level.SEVERE, null, ex);
        }

    }

    public static void main(String args[]) {
        ClientApp clientApp = new ClientApp();
        clientApp.start();

        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
        String input = "";

        try {
            do {
                input = bufferedReader.readLine();
                if (!input.equals("exit")) {
                    clientApp.session.getBasicRemote().sendText(input);
                }

            } while (!input.equals("exit"));
        } catch (IOException ex) {
            Logger.getLogger(ClientApp.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```