WEBSOCKETS CON WILDFLY (SERVIDOR) Y JAVASCRIPT (CLIENTE)

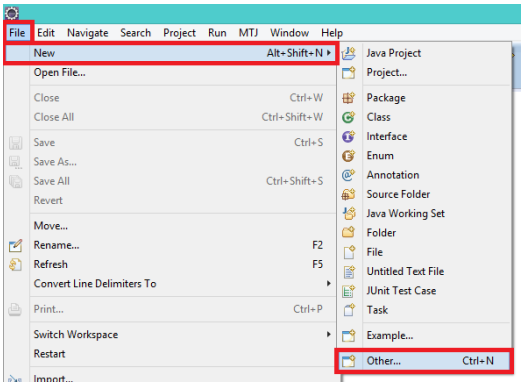
Profesor: Ing. Juan Antonio Castro Silva

Version: 1.0 (Mayo 24 de 2016)

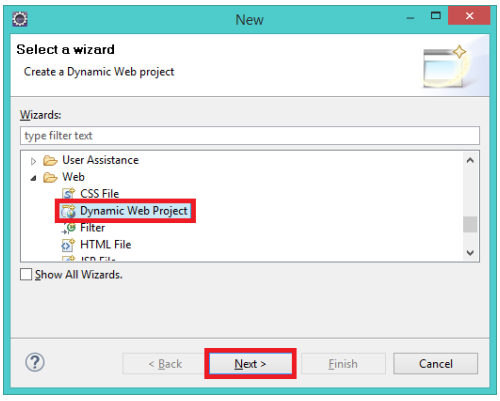
En este proyecto va a crear una aplicación web que implemente la comunicación en tiempo real con el servidor de aplicaciones wildfly y un cliente web con javascript, empleando la tecnología de WebSockets.

WebSockets implementa una comunicación full dúplex bi-direccional entre el servidor y el cliente, lo cual producirá aplicaciones web de alto rendimiento más rápidas, más escalables y más robustas.

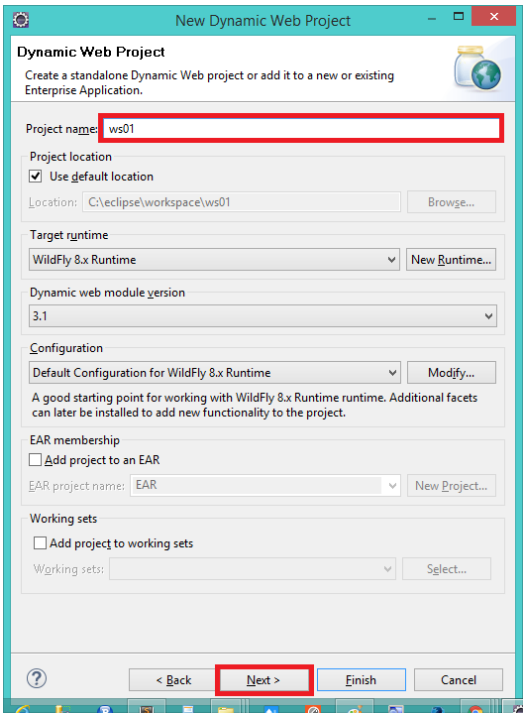
Para crear el proyecto, haga click en el menú [File], submenú [New] y finalmente en la opción [Other…].



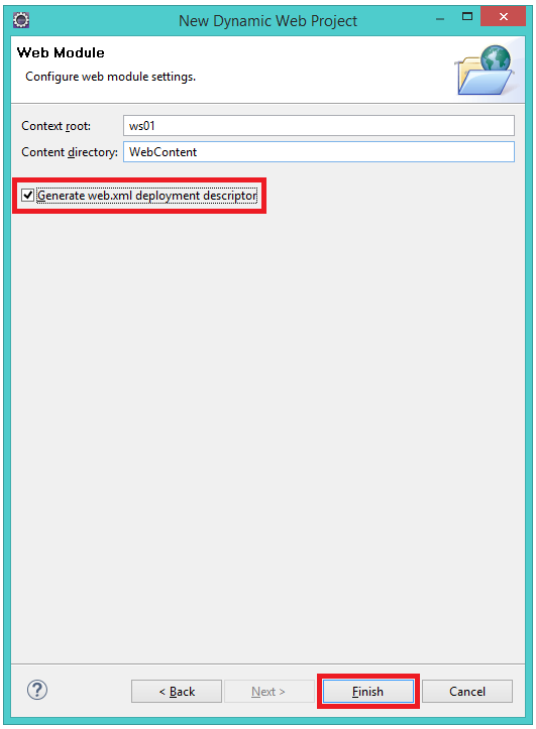
Seleccione la carpeta [Web], la opción [Dynamic Web Project] y haga click en el boton [Next].



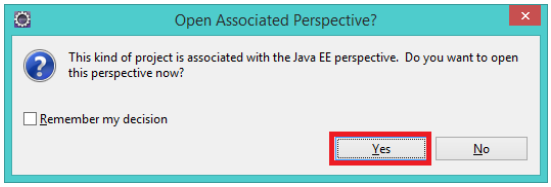
Digite el nombre del proyecto (ws01) y haga click en el boton [Next].



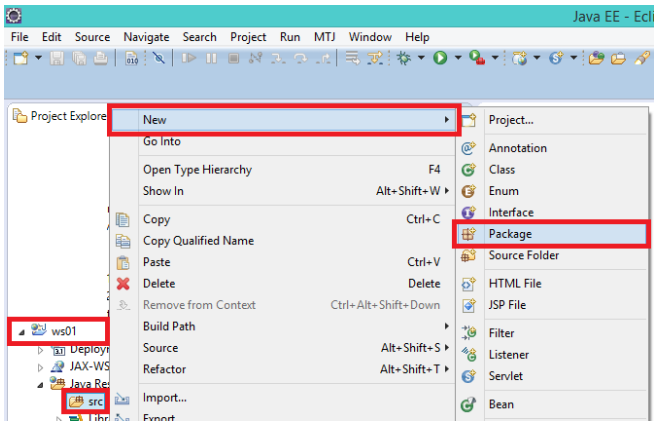
Seleccione la casilla de chequeo [Generate web.xml deployment descriptor] y haga click en el boton [Finish].



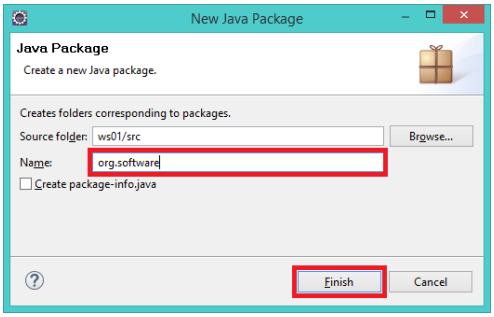
Si eclipse le solicita abrir la perspectiva de Java EE asociada al proyecto, haga click en el boton [Yes].



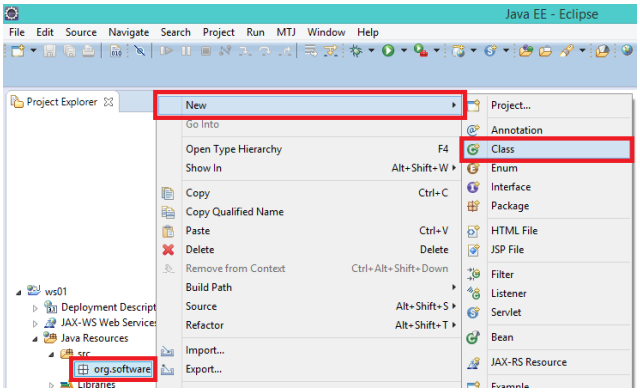
Haga click con el boton derecho del mouse sobre la carpeta [src], seleccione el menú [New] y en la opción [Package].



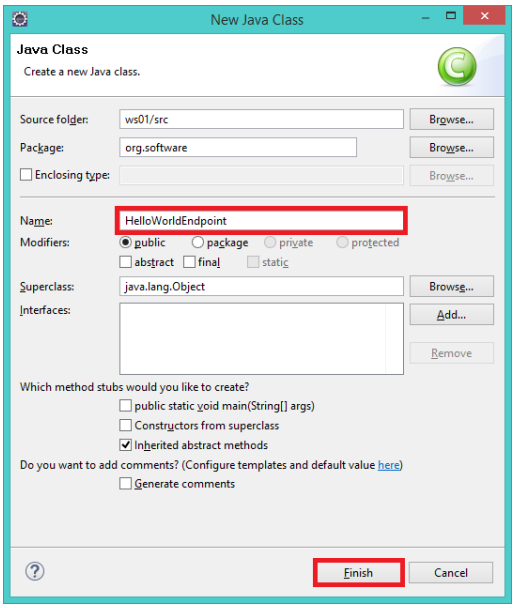
Digite el nombre del paquete (org.software) y haga click en el boton [Finish].



Para crear una clase, haga click con el boton derecho del mouse sobre el paquete [org.software], seleccione el menú [New] y la opción [Class].



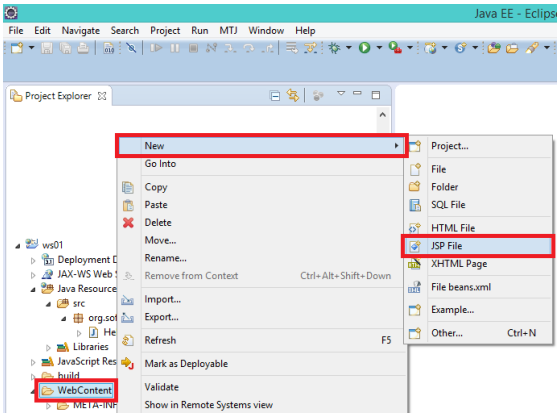
Digite el nombre de la clase (HelloWorldEndpoint) y haga click en el boton [Finish].



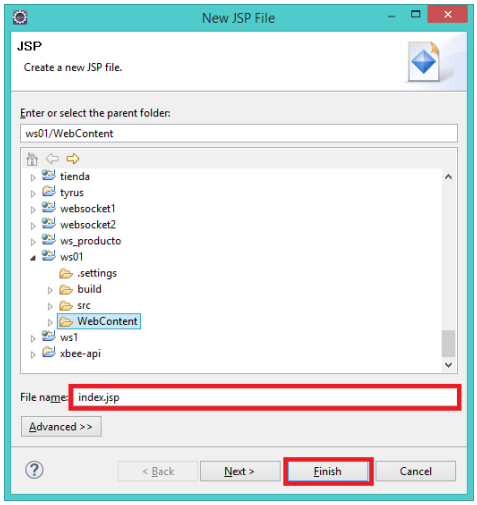
El codigo de la clase HelloWorldEndpoint (archivo HelloWorldEndpoint.java) nos permite implementar una aplicación con WebSockets.

```
package org.software;

import javax.websocket.*;
import javax.websocket.server.ServerEndpoint;

@ServerEndpoint("/hello")
public class HelloWorldEndpoint {

    @OnMessage
    public String myOnMessage(String message) {
        System.out.println("Received : "+ message);
        return message;
    }

    @OnOpen
    public void myOnOpen(Session session) {
        System.out.println("WebSocket opened: " + session.getId());
    }

    @OnClose
    public void myOnClose(CloseReason reason) {
        System.out.println("Closing a WebSocket due to " + reason.getReasonPhrase());
    }

}
```

Haga click con el boton derecho del mouse sobre la carpeta [WebContent], seleccione el menú [New] y finalmente seleccione [JSP File].



Digite el nombre de la página (index.jsp) y haga click en el boton [Finish].

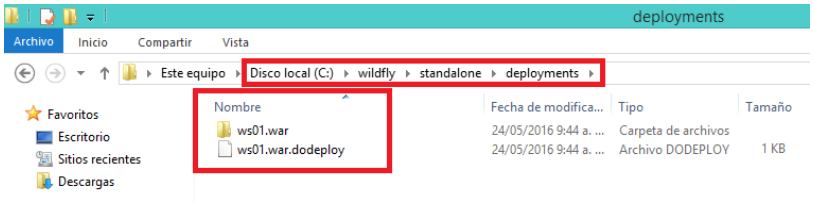


El contenido del archivo index.jsp nos permite hacer la comunicación con el servidor de aplicaciones wildfly llamando a la API JavaScript de WebSockets.

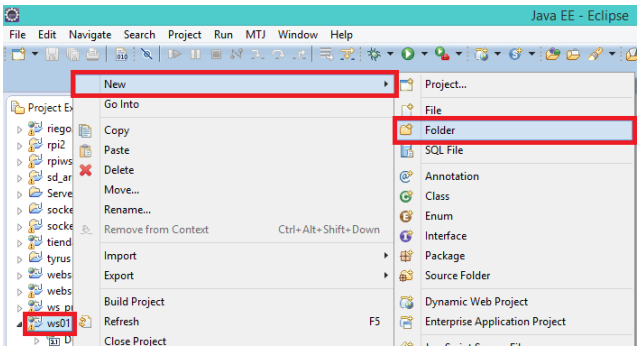
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
</head>

<body>
    <meta charset="utf-8">
    <title>HelloWorld Web sockets</title>
    <script language="javascript" type="text/javascript">
        var wsUri = getRootUri() + "/ws01/hello";

        function getRootUri() {
            return "ws://"
                + (document.location.hostname == "" ? "localhost"
                : document.location.hostname)
                + ":"
```

```
                + (document.location.port == "" ? "8080"
                : document.location.port);
        }

        function init() {
            output = document.getElementById("output");
        }

        function send_message() {
            websocket = new WebSocket(wsUri);
            websocket.onopen = function(evt) {
                onOpen(evt)
            };
            websocket.onmessage = function(evt) {
                onMessage(evt)
            };
            websocket.onerror = function(evt) {
                onError(evt)
            };
        }

        function onOpen(evt) {
            writeToScreen("Connected to Endpoint!");
            doSend(textID.value);
        }

        function onMessage(evt) {
            writeToScreen("Message Received: " + evt.data);
        }

        function onError(evt) {
            writeToScreen('<span style="color: red;">ERROR:</span> ' + evt.data);
        }

        function doSend(message) {
            writeToScreen("Message Sent: " + message);
            websocket.send(message);
        }

        function writeToScreen(message) {
            var pre = document.createElement("p");
            pre.style.wordWrap = "break-word";
            pre.innerHTML = message;

            output.appendChild(pre);
        }

        window.addEventListener("load", init, false);
    </script>

    <h2 style="text-align: center;">
        Hello World WebSocket Client
    </h2>

    <br>

    <div style="text-align: center;">
        <form action="">
            <input onclick="send_message()" value="Send" type="button">
            <input id="textID" name="message" value="Hello WebSocket!" type="text"><br>
        </form>
    </div>
    <div id="output"></div>
</body>
</html>
```

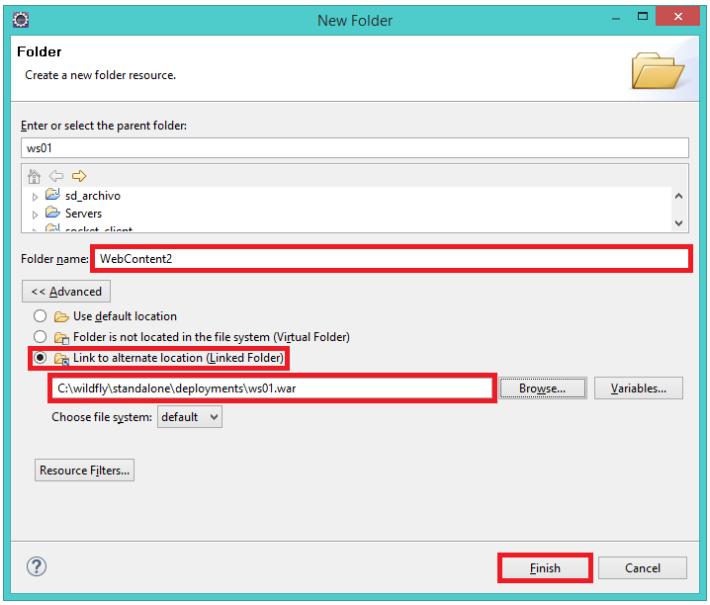
En el directorio de publicaciones (/wildfly/standalone/deployments/), cree una carpeta para almacenar los archivos de la aplicación web (ws01.war) y el archivo de publicación (ws01.war.dodeploy).



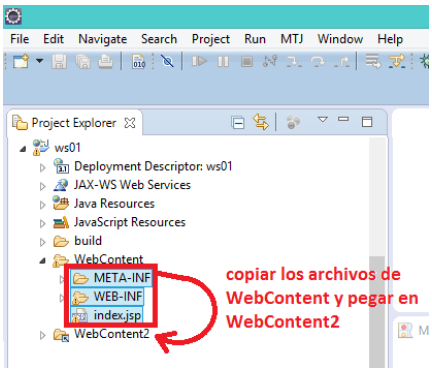
El codigo generado hasta ahora quedará en la carpeta workspace de eclipse. Para manipular directamente los archivos de la carpeta de publicaciones del wildfly cree una carpeta enlazada, haga click con el boton derecho del mouse sobre el proyecto [ws01], seleccione el menú [New] y la opción [Folder].



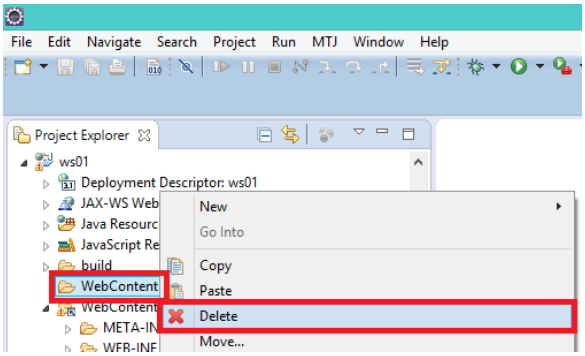
Digite el nombre de la carpeta (WebContent2), seleccione la opción [Link to alternate location (Linked Folder)], seleccione la carpeta de publicaciones de wildfly (/wildfly/standalone/deployments/) donde se alojaran los archivos de la aplicación y haga click en el boton [Finish].



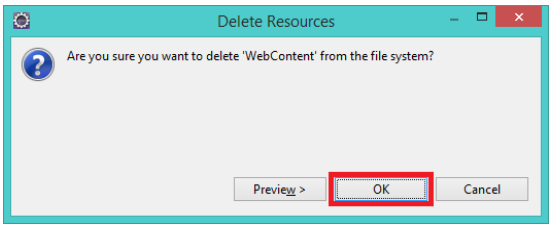
Mueva todos los archivos de la carpeta WebContent a la carpeta WebContent2.



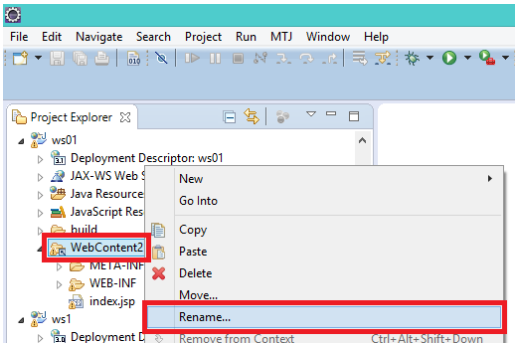
Borre la carpeta WebContent, haga click con el boton derecho del mouse sobre la carpeta WebContent y seleccione la opción [Delete].



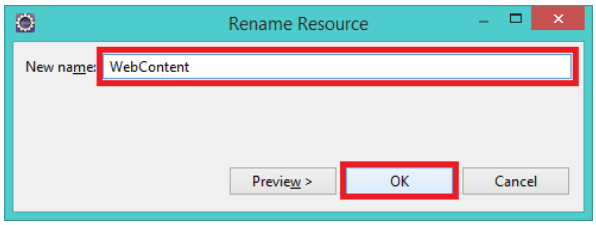
Confirme que desea borrar la carpeta WebContent, haga click en el boton [OK].



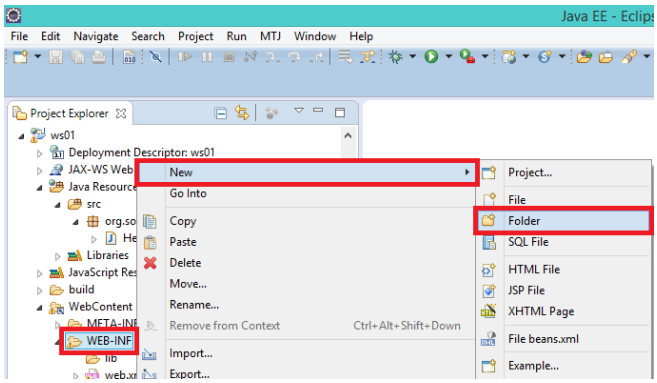
Renombre la carpeta WebContent2 como WebContent, haga click con el boton derecho del mouse sobre la carpeta WebContent2, seleccione la opción [Rename].



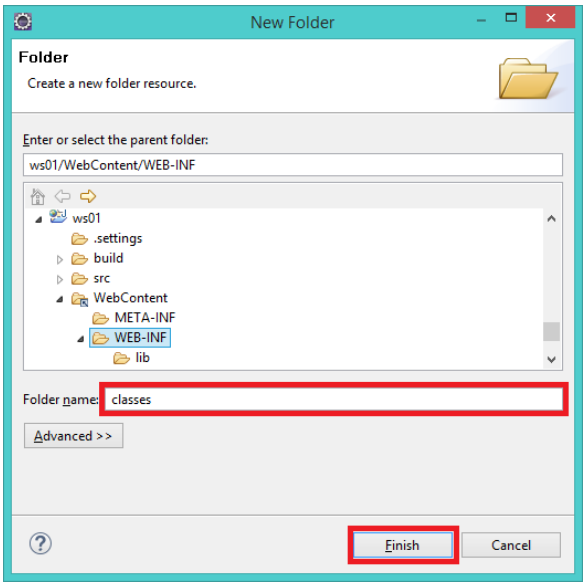
Digite el nuevo nombre de la carpeta (WebContent) y haga click en el boton [OK].



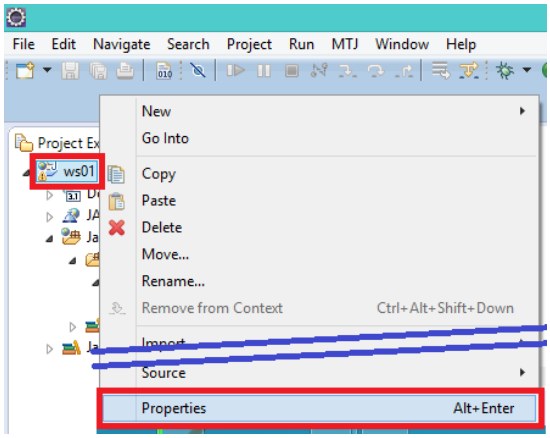
Configure el proyecto para que al compilar (construir el proyecto), el resultado se genere en la carpeta de classes del proyecto. Haga click con el boton derecho del mouse sobre la carpeta [WEB-INF], seleccione el menú [New] y finalmente la opción [Folder].



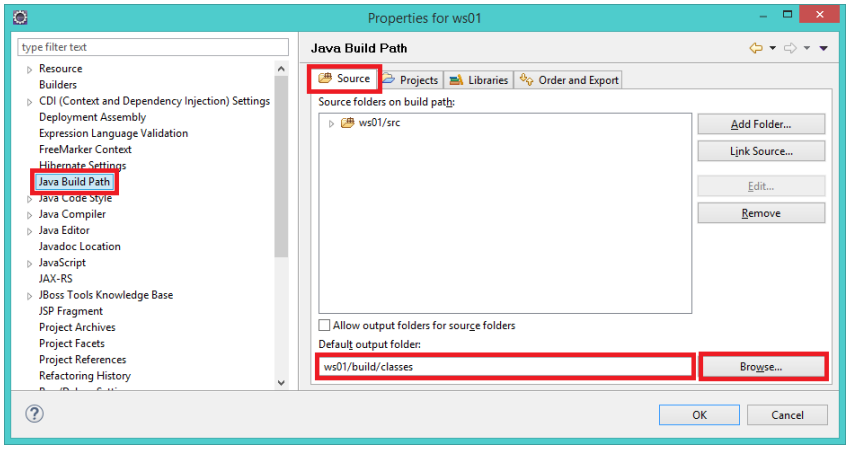
Digite el nombre de la carpeta (classes) y finalmente haga click en el boton [Finish].



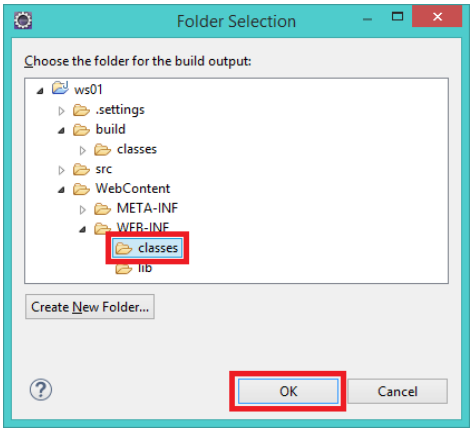
Haga click con el boton derecho del mouse sobre el proyecto y seleccione el menú [Properties].



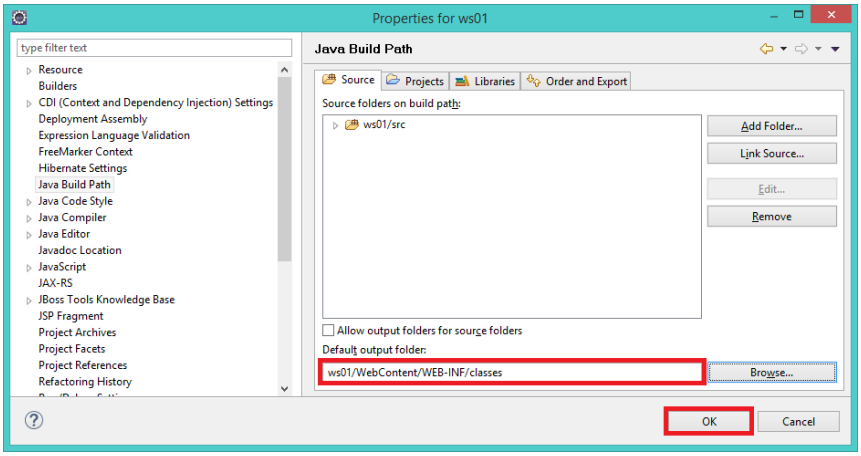
Seleccione en el menú izquierdo la opción [Java Build Path], la ficha [Source] en el centro y en la parte de abajo haga click en el boton [Browse].



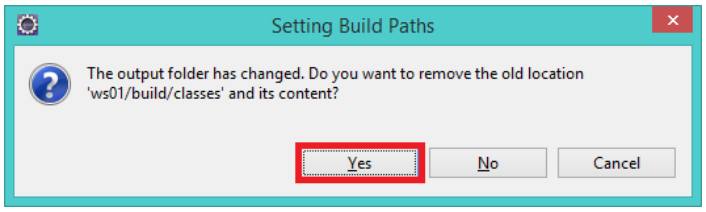
Seleccione la carpeta (classes) dentro del folder [WEB-INF] y haga click en el boton [OK].



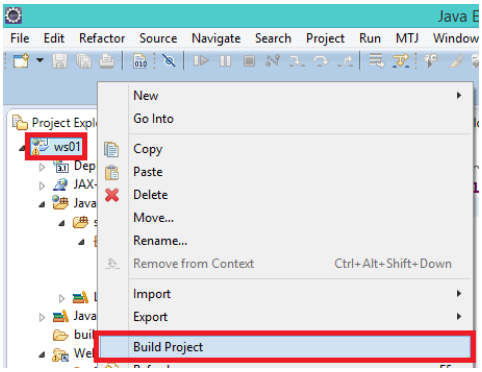
Confirme la carpeta de salida (/WebContent/WEB-INF/classes) y haga click en el boton [Finish].



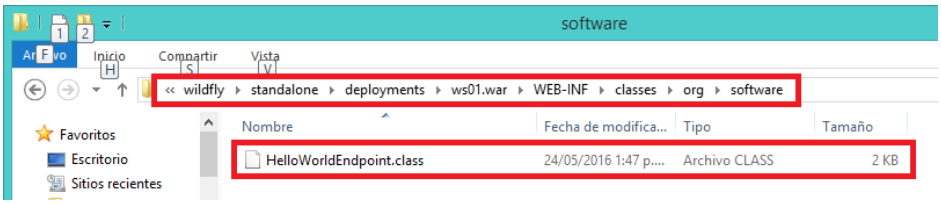
Haga click en el boton [Yes] para confirmar que desea cambiar la carpeta de salida y mover los archivos a la nueva localización.



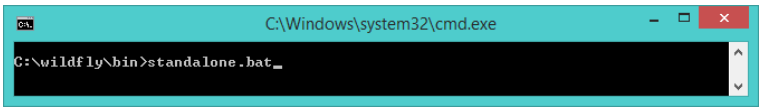
Para compilar el proyecto (Construir el proyecto), haga click con el boton derecho del mouse sobre el proyecto y seleccione la opción [Build Project].



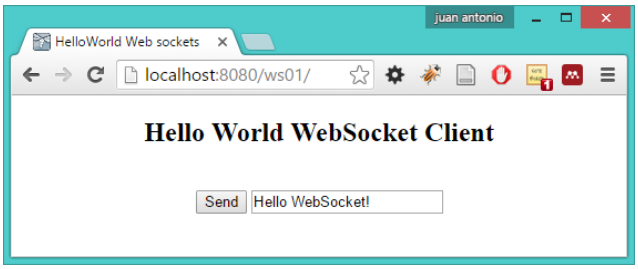
Es una buena práctica confirmar que los archivos compilados – clases (*.class) están ubicados en la carpeta de salida (WEB-INF/classes/).



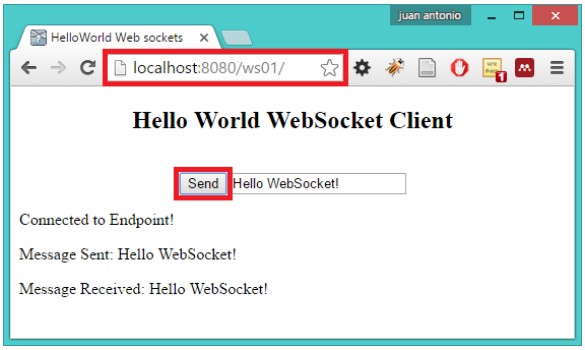
Haga correr el servidor de aplicaciones wildfly.



Abra un navegador y digite la url de la aplicación web (http://localhost:8080/ws01/).

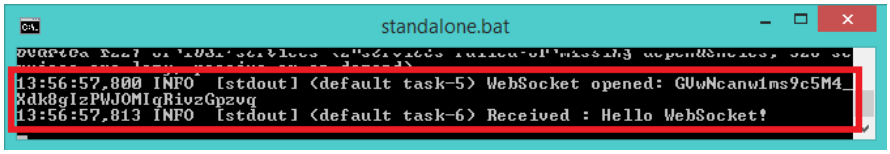


Haga click en el boton [Send] y podrá ver en la página el mensaje enviado y el recibido, demostrándose de esta forma la comunicación vía WebSockets entre el servidor de aplicaciones wildfly y el cliente JavaScript.



En la consola del wildfly podemos ver los mensajes que se mandaron a imprimir.

```
    @OnMessage
    public String myOnMessage(String message) {
        System.out.println("Received : "+ message);
        return message;
    }

    @OnOpen
    public void myOnOpen(Session session) {
        System.out.println("WebSocket opened: " + session.getId());
    }
```



El método `MyOnMessage(String message)` se emplea tanto para enviar como para recibir los mensajes.