1. Configuração do pom.xml

No arquivo pom.xml, adicione as dependências para JAX-WS:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.exemplo</groupId>
   <artifactId>JavaRPC-WebService</artifactId>
   <version>1.0-SNAPSHOT</version>

   <dependencies>
      <!-- Dependência para JAX-WS -->
      <dependency>
         <groupId>javax.jws</groupId>
         <artifactId>javax.jws-api</artifactId>
         <version>1.1</version>
      </dependency>
   </dependencies>

   <build>
      <plugins>
         <!-- Plugin para criar o JAR com dependências -->
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-assembly-plugin</artifactId>
            <version>3.3.0</version>
            <configuration>
               <descriptorRefs>
                  <descriptorRef>jar-with-dependencies</descriptorRef>
               </descriptorRefs>
            </configuration>
            <executions>
               <execution>
                  <id>make-assembly</id>
                  <phase>package</phase>
                  <goals>
                     <goal>single</goal>
                  </goals>
               </execution>
            </executions>
         </plugin>
      </plugins>
   </build>
</project>
```

2. Implementação do Código

2.1 Interface do Serviço
Crie a interface HelloService.java em src/main/java/com/exemplo:

```
package com.exemplo;

import javax.jws.WebMethod;
import javax.jws.WebService;

// Define o serviço web
@WebService
public interface HelloService {
    @WebMethod
    String sayHello(String name);
}
```

2.2 Implementação do Serviço

Crie a implementação HelloServiceImpl.java:

```
package com.exemplo;

import javax.jws.WebService;

// Implementa o serviço web
@WebService(endpointInterface = "com.exemplo.HelloService")
public class HelloServiceImpl implements HelloService {
    @Override
    public String sayHello(String name) {
        return "Olá, " + name + "! Bem-vindo ao Web Service SOAP.";
    }
}
```

2.3 Servidor
Crie a classe Server.java, que publica o Web Service:

```
package com.exemplo;

import javax.xml.ws.Endpoint;

public class Server {
    public static void main(String[] args) {
        String url = "http://localhost:8080/hello";
        System.out.println("Publicando serviço em: " + url);

        // Publica o serviço
        Endpoint.publish(url, new HelloServiceImpl());

        System.out.println("Serviço Web pronto!");
    }
}
```

2.4 Cliente
Crie a classe Client.java, que consome o Web Service:

```
package com.exemplo;

import javax.xml.namespace.QName;
import javax.xml.ws.Service;
import java.net.URL;

public class Client {
    public static void main(String[] args) {
        try {
            // URL do WSDL
            URL wsdlURL = new URL("http://localhost:8080/hello?wsdl");

            // Namespace e nome do serviço
            QName qname = new QName("http://exemplo.com/", "HelloServiceImplService");

            // Criação do serviço
            Service service = Service.create(wsdlURL, qname);
            HelloService hello = service.getPort(HelloService.class);

            // Chamada ao método remoto
            String response = hello.sayHello("Usuário");
            System.out.println("Resposta do servidor: " + response);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

3. Compilar e Executar
3.1 Compilar o Projeto

```
mvn clean package
```

3.2 Iniciar o Servidor, em um terminal, execute:

```
java -cp target/JavaRPC-WebService-1.0-SNAPSHOT-jar-with-dependencies.jar
com.exemplo.Server
```

O serviço estará disponível em: http://localhost:8080/hello?wsdl

3.3 Rodar o Cliente, em outro terminal, execute:

```
java -cp target/JavaRPC-WebService-1.0-SNAPSHOT-jar-with-dependencies.jar
com.exemplo.Client
```

4. Como Funciona
O servidor publica um Web Service SOAP em http://localhost:8080/hello.
O cliente acessa o WSDL e obtém as definições do serviço.
O cliente chama o método sayHello("Usuário") remotamente.
O servidor responde com "Olá, Usuário! Bem-vindo ao Web Service SOAP.".