**Creating a Contract-Last SOAP Service and Client Using Apache CXF, Maven, and Git in Eclipse**

**Running Tomcat**

# Background

This guide focuses mainly on the CXF aspects of building a SOAP service and client and assumes that eclipse is already up and running with a Tomcat instance and Git support. Moreover, for the approach outlined herein there is no CXF plug-in needed for eclipse or Maven; however, Spring is required. The example provided is a simple greeting service that accepts no input but simply returns a String message.

# Environment

The environment for this setup is as follows:

JRE 1.8.0\_66

Apache Tomcat 8.0 (Works with JRE 1.7 and 1.8)

eclipse MARS.2 (Maven equipped out-of-the-box)

Apache CXF 3.1.6,

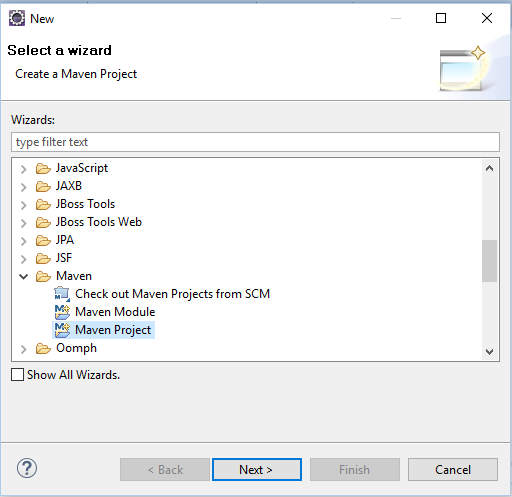
Spring 4.2.3,

Git,

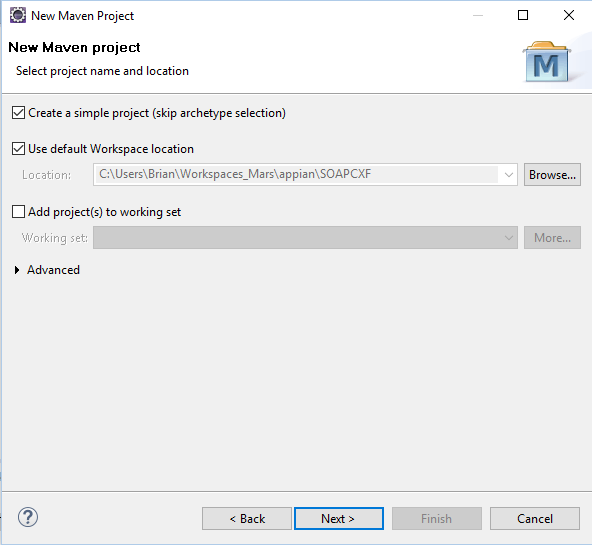
Maven 3.3

# Create a Simple SOAP Service with CXF

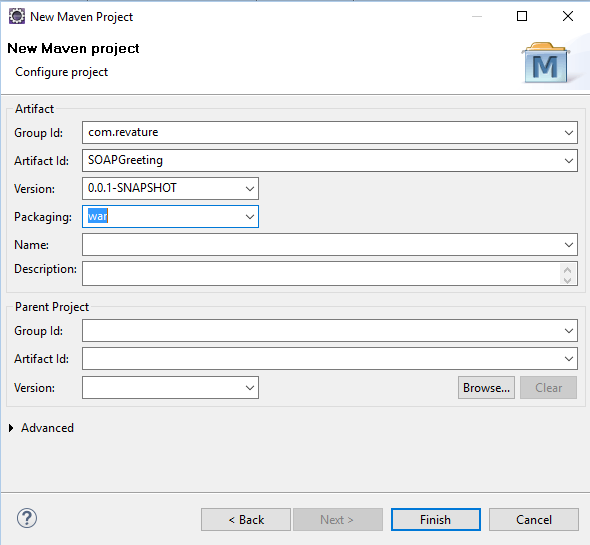
1. **New**🡪**Other…**
2. **Under the Maven folder select the Maven Project and hit Next**



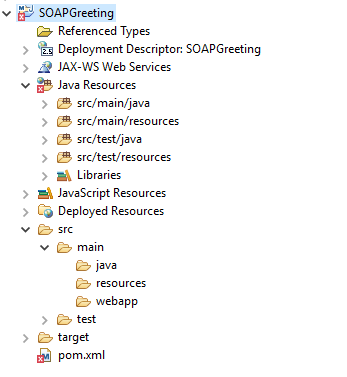
1. **Check the first two boxes as shown and hit Next:**



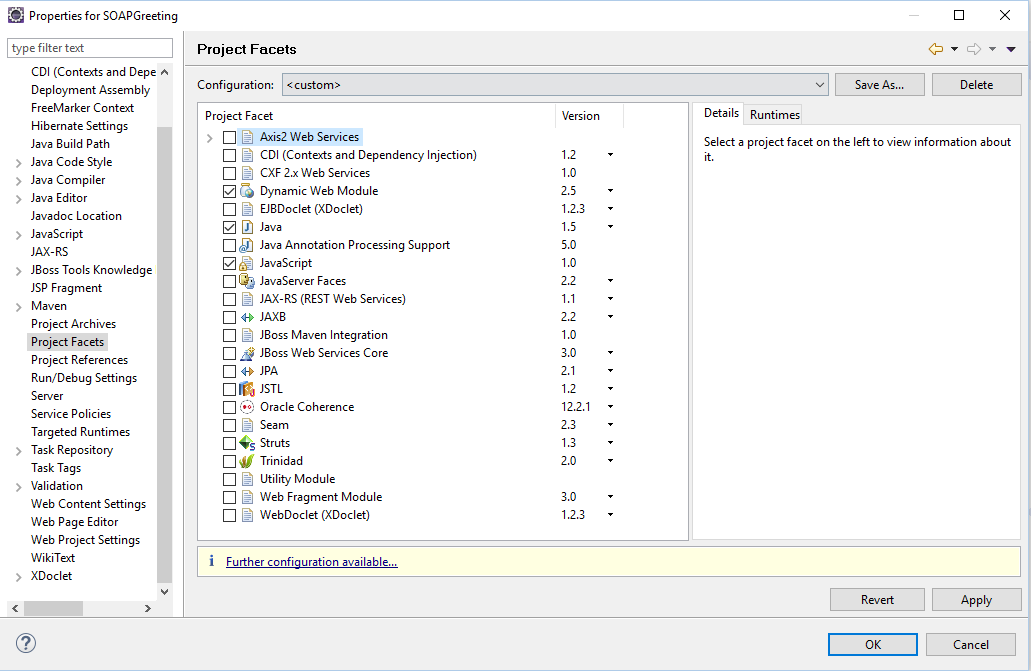
1. **Fill-in the required fields for the Maven coordinates. Note, the Artifact ID will become the project name in eclipse. Change the Packaging field to “war”.**



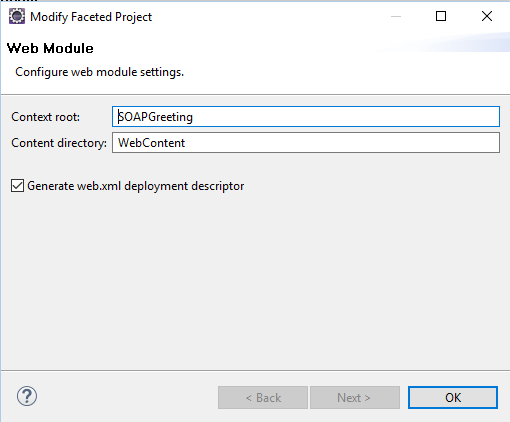
1. **The project should look like this after creation. Note the red x’s present.**



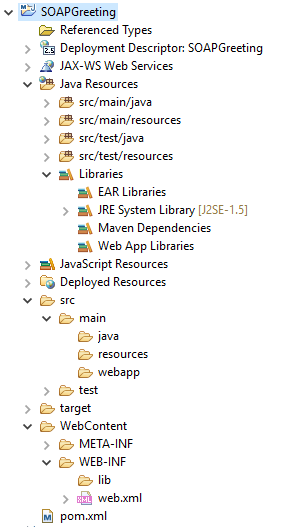
1. **Right-click on the project and select “Targeted Runtimes” and check the box: “Apache Tomcat v8.0”.**
2. **Change the Servlet container and add a web.xml by right-clicking on the project and selecting “Project Facets”. Uncheck the box: “Dynamic Web Module” and hit “Apply”. Now check the box: “Dynamic Web Module” again and choose the desired servlet container.**



1. **Touch the link: “Further configuration available…” at the bottom of the window and check the box: “Generate web.xml deployment descriptor.” Hit “OK” and then “OK” again in the next window. Note, even if you are using one of the newer annotation-based servlet containers like 3.x consider including a DD anyway so that you can easily point to the CXF Servlet and the Spring ContextLoader Listener too.**



**The project folder structure should now look like this with no red x’s present.**



1. **Double-click on the pom.xml file which is virtually empty and use the following:**

<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.revature</groupId>

<artifactId>SOAPGreeting</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<properties>

<cxf.version>3.1.6</cxf.version>

<org.springframework.version>4.2.3.RELEASE</org.springframework.version>

</properties>

<dependencies>

<!-- Web application development utilities applicable to both Servlet and

Portlet Environments(depends on spring-core, spring-beans, spring-context)Define this if you use Spring MVC, or wish to use Struts, JSF, or another web framework with Spring (org.springframework.web.\*)-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!-- apache cxf -->

<dependency>

<groupId>org.apache.cxf</groupId>

<artifactId>cxf-rt-frontend-jaxws</artifactId>

<version>${cxf.version}</version>

</dependency>

<dependency>

<groupId>org.apache.cxf</groupId>

<artifactId>cxf-rt-transports-http</artifactId>

<version>${cxf.version}</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.3</version>

<configuration>

<source>1.7</source>

<target>1.7</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**The spring-web library is necessary to find the ContextLoader Listener class defined in web.xml. Maven will automatically find all of the Spring libraries required by spring-web.**

1. **Double-click on the web.xml which is virtually empty and use the following:**

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

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns=*"http://java.sun.com/xml/ns/javaee"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"* id=*"WebApp\_ID"* version=*"2.5"*>

<servlet>

<servlet-name>CXFServlet</servlet-name>

<servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>

<load-on-startup>2</load-on-startup>

</servlet>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/beans.xml</param-value>

</context-param>

<servlet-mapping>

<servlet-name>CXFServlet</servlet-name>

<url-pattern>/\*</url-pattern>

</servlet-mapping>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

</web-app>

**Notice that we have defined the Apache CXF servlet here to service all requests. The Servlet Context ultimately tells the Spring Container were the beans.xml file is located so that it can be loaded by the listener class.**

1. **Create beans.xml in WEB-INF to define the endpoint characteristics**

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

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:jaxws=*"http://cxf.apache.org/jaxws"*

xmlns:jaxrs=*"http://cxf.apache.org/jaxrs"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://cxf.apache.org/jaxrs*

*http://cxf.apache.org/schemas/jaxrs.xsd*

*http://cxf.apache.org/jaxws*

*http://cxf.apache.org/schemas/jaxws.xsd"*>

<import resource=*"classpath:META-INF/cxf/cxf.xml"* />

<jaxws:endpoint

id=*"messageGenService"*

implementor=*"com.revature.ws.MessageGenImpl"*

address=*"/greetingservice"* />

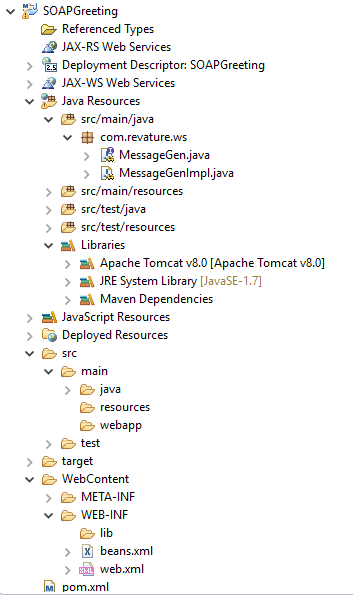
</beans>

**In the jaxws endpoint tag, the id attribute is a logical name that will be used in the annotation for the web service implementation. The implementor attribute points to the full qualified name for this implementation class. The address attribute is the name that you wish to use in the endpoint address for your web service. In this case it would be: http://localhost:8080/<project name>/<address> for a Tomcat deployment.**

1. **Update the project by right-clicking on it and selecting Maven🡪Update Project… This will re-build the project using the new pom.xml. Update the project anytime you change the pom.xml.**

**Look under Libraries for the dependencies found by Maven for this project. These are the JARS that would have to be placed on the application’s classpath (inside WEB-INF/lib) in the case of an ANT build. Without Maven, YOU would have been responsible for finding ALL of these JARS.**

1. **Create the interface/implementation pair that will provide the service for the CXF servlet endpoint. Create the service package under the Java Resources folder inside sub folder: src/main/java. The following package structure is for the beans.xml example above:**



**MessageGen.java annotated with JAX-WS**

**package** com.revature.ws;

**import** javax.jws.WebMethod;

**import** javax.jws.WebService;

@WebService

**public** **interface** MessageGen {

@WebMethod

String getMessage();

}

**MessageGenImpl.java annotated with JAX-WS. Note, the serviceName attribute comes from beans.xml.**

**package** com.revature.ws;

**import** javax.jws.WebService;

@WebService(endpointInterface = "com.revature.ws.MessageGen",

serviceName="messageGenService")

**public** **class** MessageGenImpl **implements** MessageGen{

**public** String getMessage() {

**return** "Hi Brian...";

}

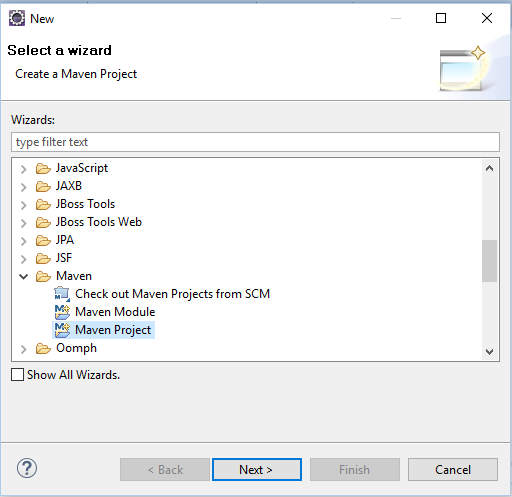
}

1. **Deploy your SOAP service on Tomcat like you would a web app and go to:**

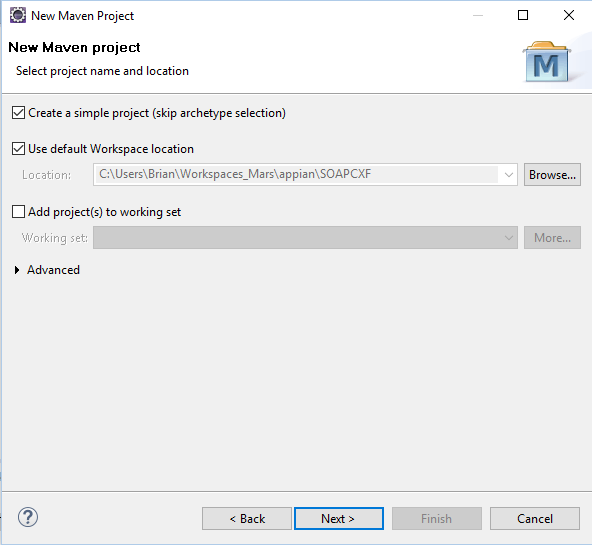
[**http://localhost:8080/SOAPGreeting/greetingservice?wsdl**](http://localhost:8080/SOAPGreeting/greetingservice?wsdl) **to see the WSDL file.**

# Create a SOAP Client with CXF

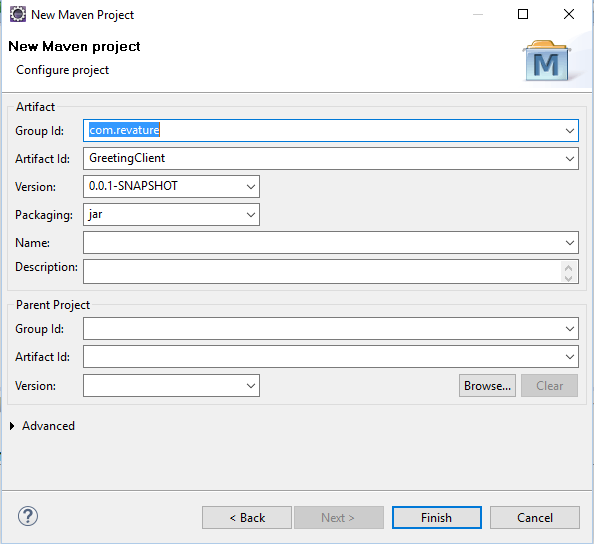
1. **New**🡪**Other…**
2. **Under the Maven folder select the Maven Project and hit Next**



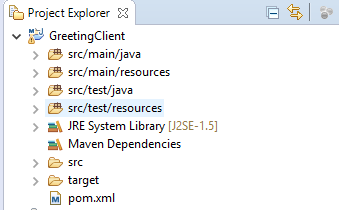
1. **Check the first two boxes as shown and hit Next:**



1. **Fill-in the required fields for the Maven coordinates. Note, the Artifact ID will become the project name in eclipse.**



1. **The project should look like this after creation.**



1. **Make sure the SOAP service to be consumed is up and running or errors will appear inside the POM.**
2. **Double-click on the pom.xml file which is virtually empty and use the following below. This same POM can be used *inside* eclipse for any SOAP service. Simply change the URL of the deployed WSDL in the wsdl tags below.**

<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/maven-v4\_0\_0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.revature</groupId>

<artifactId>greeting-client</artifactId>

<version>1.0</version>

<name>SOAP greeting client</name>

<packaging>jar</packaging>

<properties>

<cxf.version>3.1.6</cxf.version>

</properties>

<dependencies>

<!-- apache cxf -->

<dependency>

<groupId>org.apache.cxf</groupId>

<artifactId>cxf-rt-frontend-jaxws</artifactId>

<version>${cxf.version}</version>

</dependency>

<dependency>

<groupId>org.apache.cxf</groupId>

<artifactId>cxf-rt-transports-http</artifactId>

<version>${cxf.version}</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-eclipse-plugin</artifactId>

<configuration>

<projectNameTemplate>[artifactId]-[version]</projectNameTemplate>

<wtpmanifest>true</wtpmanifest>

<wtpapplicationxml>true</wtpapplicationxml>

<wtpversion>2.0</wtpversion>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.cxf</groupId>

<artifactId>cxf-codegen-plugin</artifactId>

<version>${cxf.version}</version>

<executions>

<execution>

<phase>generate-sources</phase>

<goals>

<goal>wsdl2java</goal>

</goals>

<configuration> <sourceRoot>${basedir}/src/main/java</sourceRoot>

<wsdlOptions>

<wsdlOption> <wsdl>http://localhost:8080/SOAPGreeting/greetingservice?wsdl</wsdl>

<extraargs>

<extraarg>-client</extraarg>

</extraargs>

</wsdlOption>

</wsdlOptions>

</configuration>

</execution>

</executions>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.7</source>

<target>1.7</target>

</configuration>

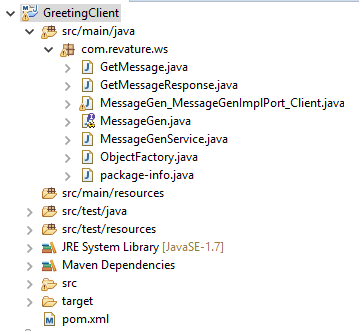
</plugin>

</plugins>

</build>

</project>

1. **Right-click on the client project and select Maven🡪Update Project… The project should now look like this. CXF has built all the binding and support classes for you. Notice also, that the JRE version now matches the compiler setting in the POM.**



1. **Create the client main class and consume the service according to the WSDL.**

**package** com.revature.ws;

**public** **class** MessageMain {

**public** **static** **void** main(String[] args) {

MessageGenService msgen = **new** MessageGenService();

MessageGen msport = msgen.getMessageGenImplPort();

System.***out***.println(msport.getMessage());

}

}

# Viewing a SOAP Message with JAX-WS

# WSDL for Greeting Service

**<wsdl:definitions** xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://ws.revature.com/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"xmlns:ns1="http://schemas.xmlsoap.org/soap/http" name="messageGenService" targetNamespace="http://ws.revature.com/">

**<wsdl:types>**

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://ws.revature.com/" elementFormDefault="unqualified" targetNamespace="http://ws.revature.com/" version="1.0">

<xs:element name="getMessage" type="tns:getMessage"/>

<xs:element name="getMessageResponse" type="tns:getMessageResponse"/>

<xs:complexType name="getMessage">

<xs:sequence/>

</xs:complexType>

<xs:complexType name="getMessageResponse">

<xs:sequence>

<xs:element minOccurs="0" name="return" type="xs:string"/>

</xs:sequence>

</xs:complexType>

</xs:schema>

**</wsdl:types>**

**<wsdl:message name="getMessageResponse">**

<wsdl:part element="tns:getMessageResponse" name="parameters"></wsdl:part>

</wsdl:message>

**<wsdl:message name="getMessage">**

<wsdl:part element="tns:getMessage" name="parameters"></wsdl:part>

</wsdl:message>

**<wsdl:portType** name="MessageGen">

<wsdl:operation name="getMessage">

<wsdl:input message="tns:getMessage" name="getMessage"></wsdl:input>

<wsdl:output message="tns:getMessageResponse" name="getMessageResponse"></wsdl:output>

</wsdl:operation>

**</wsdl:portType>**

**<wsdl:binding** name="messageGenServiceSoapBinding" type="tns:MessageGen">

<soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

<wsdl:operation name="getMessage">

<soap:operation soapAction="" style="document"/>

<wsdl:input name="getMessage">

<soap:body use="literal"/>

</wsdl:input>

<wsdl:output name="getMessageResponse">

<soap:body use="literal"/>

</wsdl:output>

</wsdl:operation>

**</wsdl:binding>**

**<wsdl:service** name="messageGenService">

<wsdl:port binding="tns:messageGenServiceSoapBinding" name="MessageGenImplPort">

<soap:address location="http://localhost:8080/SOAPGreeting/greetingservice"/>

</wsdl:port>

**</wsdl:service>**

**</wsdl:definitions>**