# Getting NREL Source

* mkdir –p /usr/local/git
* cd /usr/local/git
* git clone <https://github.com/sixthcolumn/nrel.git>
* cd /usr/local/git/nrel

# Adding a new service to axis2

For this example, we are creating new service **fooservice.wsdl**

* cd /usr/local/git/nrel/61968
* cp fooservice.wsdl wsdl/
* vi pom.xml
* add a new entry in the wsdl generation section :

<plugin>

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

<artifactId>cxf-codegen-plugin</artifactId> <version>${cxf.version}</version>

<executions>

<execution>

<id>generate-sources</id>

<phase>generate-sources</phase>

<configuration>

<wsdlOption>

<wsdl>${basedir}/wsdl/ExecuteFooservice.wsdl</wsdl>

<extraargs>

<extraarg>-client</extraarg>

</extraargs>

</wsdlOption>

**<wsdlOption>**

**<wsdl>${basedir}/wsdl/fooservice.wsdl</wsdl>**

**<extraargs>**

**<extraarg>-client</extraarg>**

**</extraargs>**

**</wsdlOption>**

</configuration>

<goals>

<goal>wsdl2java</goal>

</goals>

</execution>

</executions>

</plugin>

# Generating/Implementing Source

This project uses maven to build the war file. In order to create the new fooservice, we must generate the axis2 source, then implement our own java code, which will implement the stab class created by axis2. This is a Spring project, and thus, multiple configuration files must be modified. And lastly, the developer writes code to implement customer specific logic on fooservice.

## Source Generation

* mvn genereate-sources
* the above creates and populates the directory target/generated-sources
  + find the java source for your service under this directory.
  + for examples, for dispatch the service filed is : com.epri.executeFooservice.FooservicePort;
  + most likely your axis2 skeleton is named fooservicePort

## Create Business Logic Source File

This document assumes you are using Eclipse as your editor.

* Start Eclipse
* Import source file directory /usr/local/git/nrel/61968
* New Java source file : com.sixthc.server.ws.executeFoo.java
* In ExecuteFoo.java, have it implement FooservicePort
* Example file :

**package** com.sixthc.server.ws;

**import** java.util.UUID;

**import** javax.xml.ws.Holder;

**import** org.apache.log4j.Logger;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** ch.iec.tc57.\_2011.schema.message.HeaderType;

**import** ch.iec.tc57.\_2011.schema.message.ReplyType;

**import** ch.iec.tc57.\_2011.schema.message.RequestType;

**import** com.epri.Fooservicemessage.FoorservicePayloadType;

**import** com.epri.executeFooservice.FooservicePort;

**import** com.sixthc.dao.MessageLogDao;

**import** com.sixthc.interceptor.CIMLoggingInInterceptor;

**public** **class** ExecuteFoo **implements** FooservicePort {

**private** **static** org.apache.log4j.Logger *log* = Logger

.*getLogger*(ExecuteFooservice.**class**);

@Override

**public** **void** createFoo(Holder<HeaderType> header,

RequestType request, Holder<FooservicePayloadType> payload,

Holder<ReplyType> reply)

**throws** com.epri.executeFooservice.FaultMessage {

reply.value = **new** ReplyType();

reply.value.setResult("OK");

header.value.setCorrelationID(header.value.getMessageID());

header.value.setMessageID(UUID.*randomUUID*().toString());

*log*.debug("createDERGroups set correlation id [" + header.value.getCorrelationID() + "]");

*log*.debug("createDERGroups set message id [" + header.value.getMessageID() + "]");

}

}

Note :

* business logic implemented in this source file
* apache logging : debug and info logging to tomcat log file

## Create Spring Configuration File

* cd 61968/src/main/webapp/WEB-INF/spring
* create file fooservice.xml

Example file for Fooservice :

<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:cxf="http://cxf.apache.org/core"

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

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

http://cxf.apache.org/core

http://cxf.apache.org/schemas/core.xsd

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

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

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

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

<bean id="executeFooBean" class="com.sixthc.server.ws.ExecuteFoo" />

<jaxws:endpoint id="executeFoo" implementor="#executeFooBean"

address="/executeFoo">

<jaxws:inInterceptors>

<bean class="com.sixthc.interceptor.CIMLoggingInInterceptor">

<property name="packageName" value="FOO DER" />

<property name="packageGroupName" value="FOO Control" />

<property name="messageName" value="FOO\_DER(createFoo)" />

<property name="resultCode" value="success" />

<property name="stage" value="serviceIn" />

<property name="strict" value="false" />

</bean>

</jaxws:inInterceptors>

<jaxws:outInterceptors>

<bean class="com.sixthc.interceptor.CIMLoggingOutInterceptor">

<property name="packageName" value="FOO DER" />

<property name="packageGroupName" value="FOO Control" />

<property name="messageName" value="FOO\_DER(createFoo) " />

<property name="resultCode" value="success" />

<property name="stage" value="serviceOut" />

<property name="strict" value="false" />

</bean>

</jaxws:outInterceptors>

<jaxws:outFaultInterceptors>

<bean class="com.sixthc.interceptor.CIMLoggingOutInterceptor">

<property name="packageName" value="FOO DER" />

<property name="packageGroupName" value="FOO Control" />

<property name="messageName" value="FOO\_DER(createDERGroupDispatch)es " />

<property name="resultCode" value="fail" />

<property name="stage" value="serviceOutFault" />

</bean>

</jaxws:outFaultInterceptors>

<jaxws:inFaultInterceptors>

<bean class="com.sixthc.interceptor.CIMLoggingInInterceptor">

<property name="packageName" value="FOO DER" />

<property name="packageGroupName" value="FOO Control" />

<property name="messageName" value="FOO\_DER(createFoo) " />

<property name="resultCode" value="fail" />

<property name="stage" value="serviceInFault" />

</bean>

</jaxws:inFaultInterceptors>

<jaxws:properties>

<entry key="schema-validation-enabled" value="true" />

<entry key="jaxb-validation-event-handler">

<bean class="com.sixthc.interceptor.DetailValidationHandler" />

</entry>

</jaxws:properties>

</jaxws:endpoint>

</beans>

Explanation of File Contents :

The spring file implements logging interceptors for the executeFoo.java file. When the class file is loaded into the war file, spring will call the logging interceptors with the package variables whenever the service is called.

* executeFooBean – bean id, used in context file
* /executeFoo – url of foo service
* packagename “FOO DER” – database configuration package name
* packageGroupName “FOO Control” – database package group name
* messageName FOO\_DER (createFoo) – database message name

## Database Configuration

In the configuration file above, three unique identifiers were created :

* package name : “FOO DER”
* package group name : “FOO Control”
* message name : “FOO\_DER (createFoo)

These three variables must be populated in the database for the purpose of logging this new type of message. Create rows in the following tables :

* insert package name table info here
* insert package group table info here
* insert message name info here

## Update Context File

* vi cd 61968/src/main/webapp/WEB-INF /cxf-contest.xml
* the line to be created references 61968/src/main/webapp/WEB-INF/spring/fooservice.xml (created in the last step)
* add the following line :

<import resource="spring/ fooservice.xml" />

## Building the war File

* cd nrel/61968
* mvn package
* cd target
* ls epriConnect.war (the tomcat war file)
* cp epriConnect.war …/tomcat/webapps

## Testing the Application

* Start soapui
* new project, use the foo.wsdl file that implemented the service
* create test to call ‘createFoo’
* point the createFoo test to the tomcat service
* execute the test, verify the catalina.out file in …/tomcat/logs/catalina.out
* verify the message was logged in the database