Sistema Integral Multicanal de Atención al Ciudadano

e-SIRCA-Manual_Generación_de_Servicios_gvNIX

GENERALITAT VALENCIANA
CONSELLERIA DE HACIENDA Y ADMINISTRACIÓN PÚBLICA
DIRECCIÓN GENERAL DE TECNOLOGÍAS DE LA INFORMACIÓN

Versión 003
Agosto de 2014

Unió Europea
Fons Europeu de Desenvolupament Regional
Una manera de fer Europa

Unión Europea
Fondo Europeo de Desarrollo Regional
Una manera de hacer Europa

Índice

1 Control del documento

Información general

Título	Manual de usuario de generación de Servicios mediante gvNIX
Creado por	DGTI
Revisado por	
Lista de distribución	
Nombre del fichero	e-SIRCA-CONSTRUCCION-Manual_Usuario_Generación_de_Servicios_gvNIX_v3.odt

Histórico de revisiones

Versión	Fecha	Autor	Observaciones
001	14/08/2014	DGTI	Versión inicial
002	21/08/2014	DGTI	Añadidos Anexo II y III
003	28/08/2014	DGTI	Añadido Anexo IV y revisión de contenido.

Estado del documento

Responsable aprobación	Fecha

Alcance

Este documento pretende ser una guía de usuario para la generación de servicios servidores mediante el framework gvNIX siguiendo el protocolo SCSP adoptado por la Plataforma Autonómica de Intermediación de Datos Segura (a partir de ahora PAI).

Objetivos

Los objetivos del presente documento son:

- Describir las particularidades y condicionantes de la generación de servicios servidores mediante gvNIX.

Audiencia

Nombre y Apellidos	Rol

Tabla 1: Audiencia

Glosario

Término	Definición
PAI	Plataforma Autonómica de Intermediación de Datos Segura

Tabla 2: Glosario

Referencias

Referencia	Título

Tabla 3: Referencias

2 Introducción

La Plataforma Autonómica de Intermediación de Datos Segura (PAI) pretende generar una guía de usuario para la generación de servicios web mediante el framework gvNIX siguiendo el protocolo SCSP.

Se ha de tener en cuenta que todo el proceso de instalación y configuración básica de los entornos queda delegado a la documentación de gvNIX, donde se relata más ampliamente. Este documento solo relata los pasos adicionales o alternativos que se han de realizar para el correcto funcionamiento de un servicio web creado con este framework.

3 Guía de para la generación de servicios web mediante gvNIX

Configuración y requisitos

El código fuente ejemplo que se comenta en esta sección se puede bajar desde el portal de difusión en la sección -¿cómo usar la plataforma?, que se encuentra en el área de la Plataforma de Interoperabilidad.

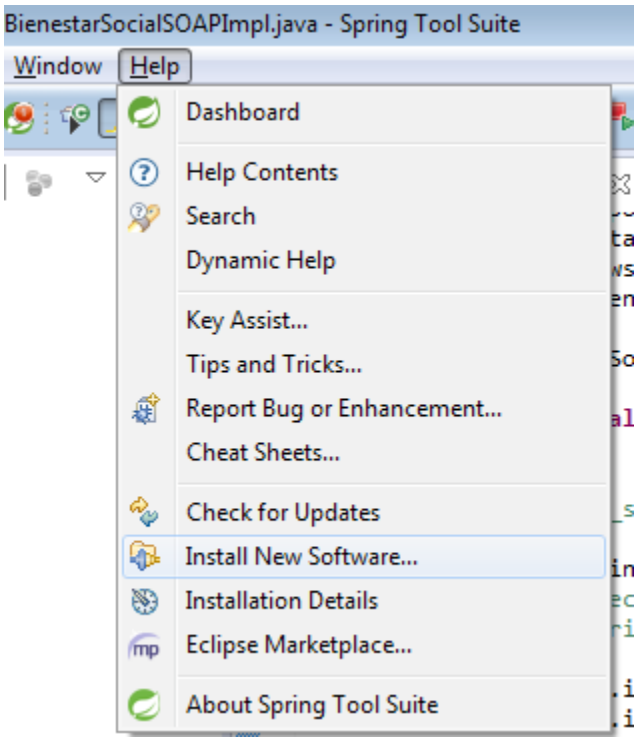
Se trata de un proyecto ejemplo en gvNIX, que contiene los fuentes y un compilado preparado para ejecutar en cualquier servidor de aplicaciones que lo permita. Se han utilizado para realizarlo y testearlo, las siguientes tecnologías:

- String Tool Suite en su versión 3.6.0
  - http://spring.io/tools
- gvNIX version 1.3.1
  - http://www.gvnix.org/
- Plugin para STS Spring Roo Support.
  - Help -> Install New Software… en STS
- SoapUI : Herramienta para testeo de WebServices, usaremos la versión 4,0,1
  - http://www.soapui.org/

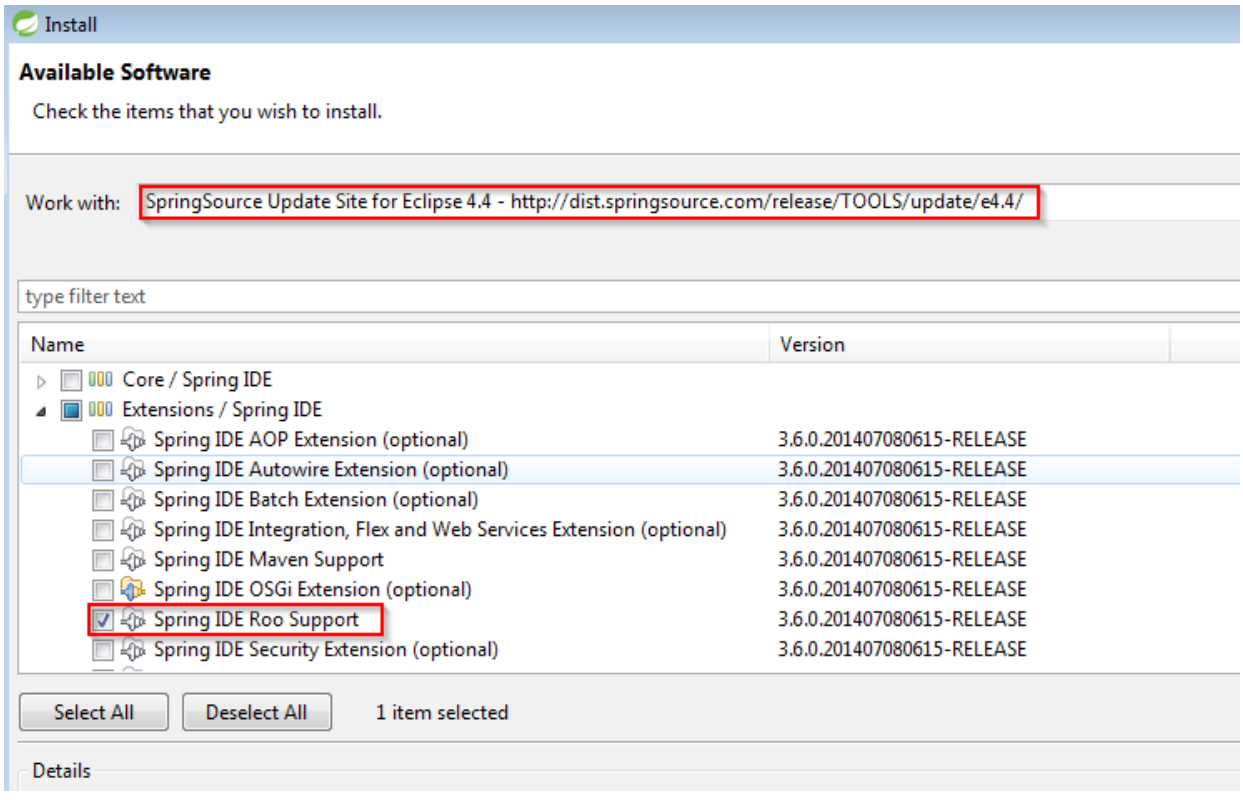
Generación de servicio web

Proyecto en Spring Tool Suite (STS)
Una vez descargado y ejecutado el STS, versión 3.6.0 desde la página http://spring.io/tools, es posible que debamos instalar el plugin 'Spring IDE Roo Support' que en la documentación gvNIX se entiende esta incluido en el IDE STS, pero en su última versión se encuentra como plugin adicional.

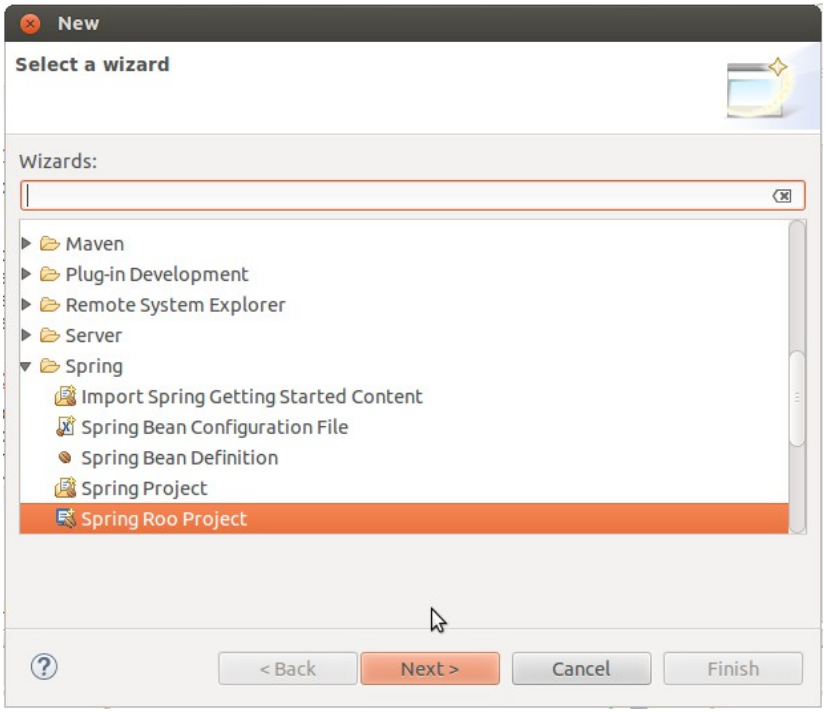
Nos vamos a Help -> Install New Software



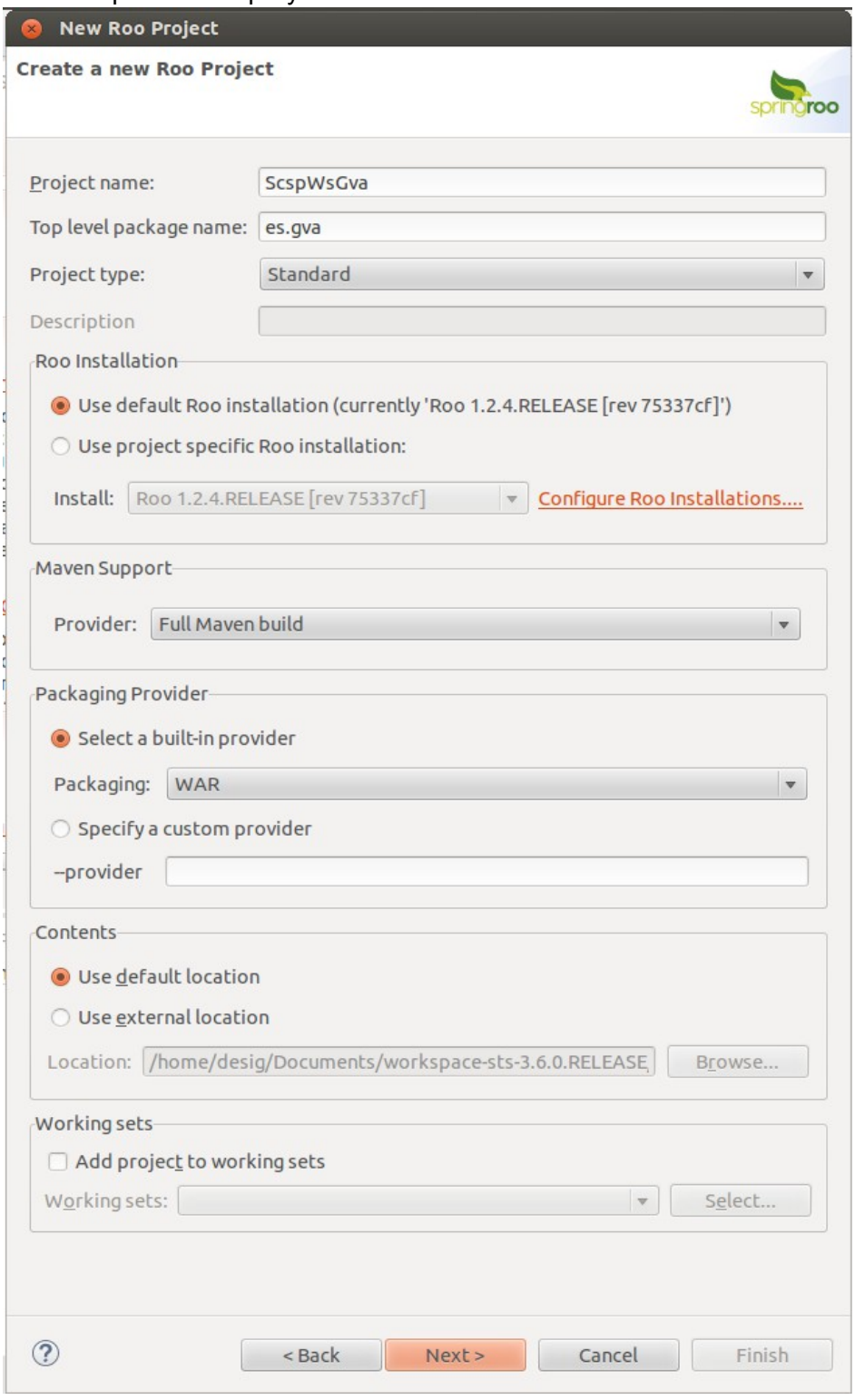
Seleccionamos el repositorio de Spring y en él Extensions / Spring IDE -> Spring IDE Roo Support



Una vez instalado y configurado el entorno, podemos realizar la creación del proyecto, desde la interfaz de STS.



Seleccionamos las opciones de proyecto.



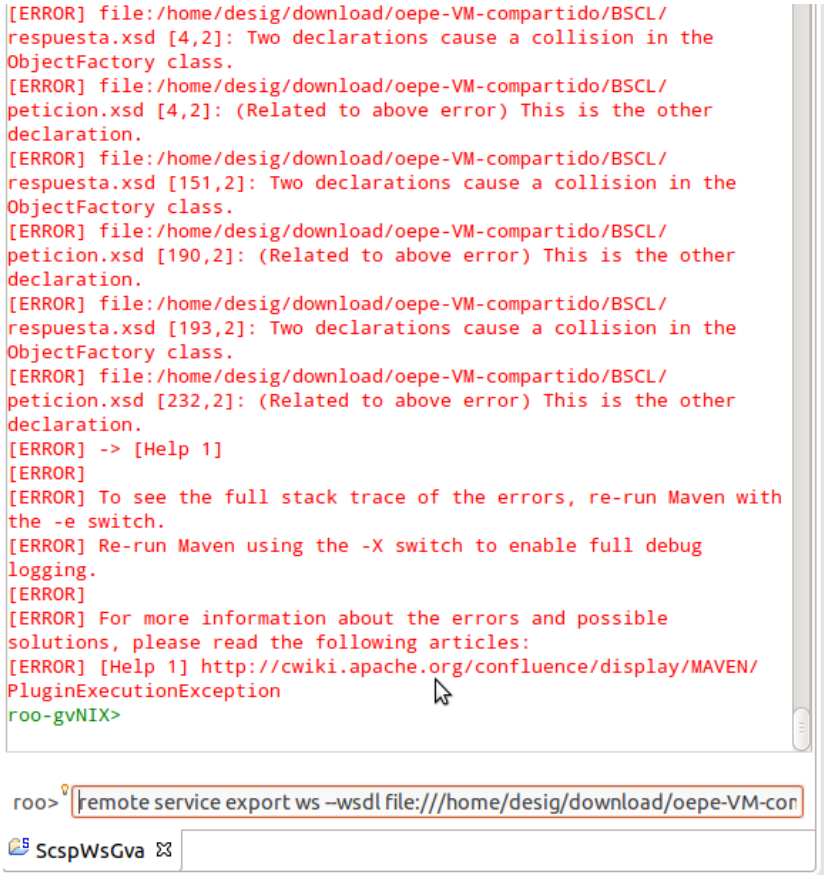
Sobre la Shell introducimos los comandos necesarios para generar un proyecto básico que cumpla nuestra necesidades. Para ello, ejecutamos la orden '***web mvc setup***'

Tras crear el proyecto y su estructura básica, debemos mediante el wsdl correcto generar el servicio web, La orden '***remote service export ws --wsdl ruta_wsdl***' es la idónea para ello, pero falla en ocasiones cuando la estructura del wsdl es compleja y existen objetos etiquetados con el mismo nombre en los diferentes xsd, como es el caso.

Este comportamiento es derivado de la asignación de un nombre de paquete específico. ***CXF*** intenta generar las clases objeto en ese momento y se produce el conflicto.

gvNix reproduce el problema conocido al usar ***CXF*** en la generación de las clases. Para resolverlo hemos de tratar de evitar especificar un paquete destino, de forma que ***CXF*** resuelva la estructura destino de los paquetes a generar. Para ello, y dado que en la versión actual del framework no está resuelto todavía, se deben hacer estos pasos de forma manual. Se exponen a continuación los pasos a seguir:

Ejecutar la orden '***remote service export ws --wsdl ruta_wsdl***' que reportará los errores comentados.



NOTA: Es necesario ejecutar la orden al menos una vez para que genere el pom correctamente aunque falle la instrucción.

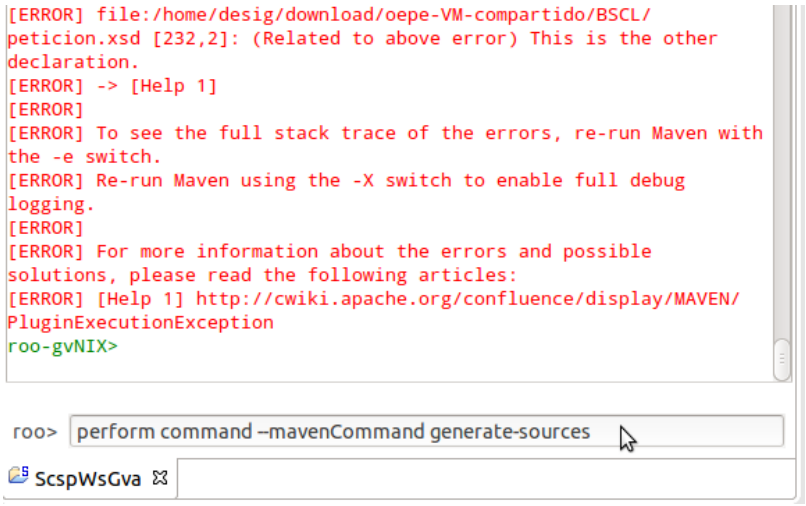
Tras ejecutar la orden anterior, el pom del proyecto se debe haber modificado incluyendo una sección de esta forma

```
<plugin>
      <groupId>org.apache.cxf</groupId>
      <artifactId>cxf-codegen-plugin</artifactId>
      <version>${cxf.version}</version>
```

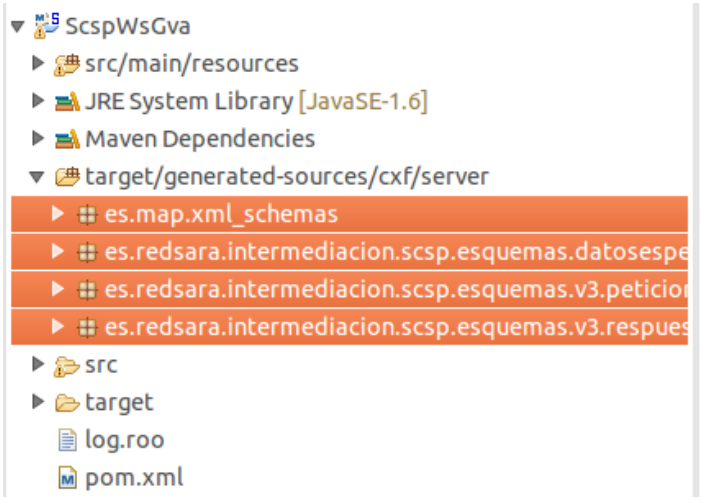
```
        <executions>
                <execution>
                        <id>generate-sources-cxf-server</id>
                        <phase>none</phase>
                        <goals>
                                <goal>wsdl2java</goal>
                        </goals>
                        <configuration>
                                <sourceRoot>${basedir}/target/generated-
sources/cxf/server</sourceRoot>
                                <defaultOptions>
                                        <extendedSoapHeaders>true</extendedSoapHeaders>
                                        <autoNameResolution>true</autoNameResolution>
                                </defaultOptions>
                                <wsdlOptions>
                                        <wsdlOption>
                                                <wsdl>///home/desig/download/oepe-VM-
compartido/BSCL/BienestarSocial.wsdl</wsdl>
                                                <extraargs>
                                                        <extraarg>-impl</extraarg>
                                                </extraargs>
                                                <!--packagenames>
                                                 <packagename>es.map.www.xmluschemas</packagename>
                                            </packagenames-->
                                        </wsdlOption>
                                </wsdlOptions>
                        </configuration>
                </execution>
        </executions>
</plugin>
```

Debemos modificar en la sección <phase> y cambiar '***none***' por '***generate-sources***' y comentar la sección '***packagenames***' para que el plugin codegen no genere el mismo error comentado.

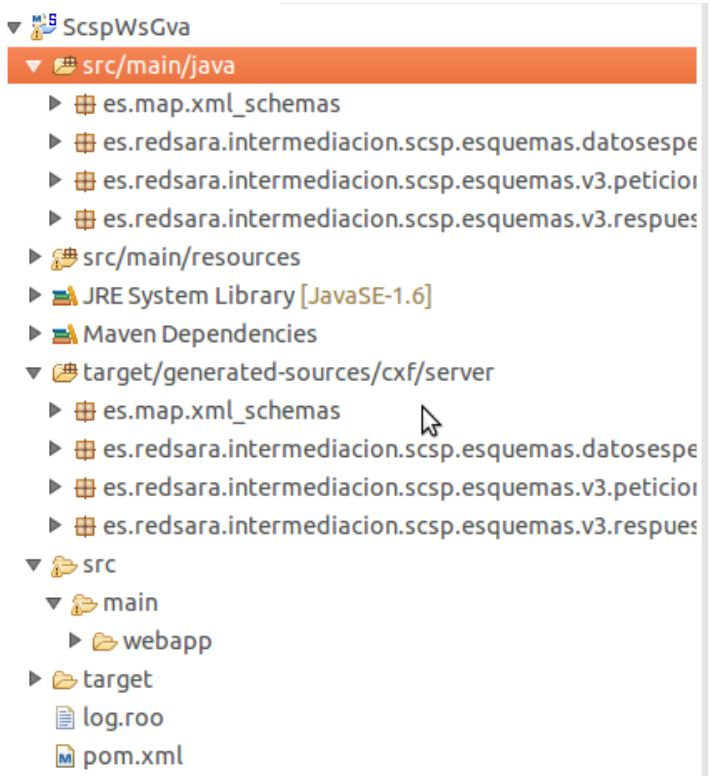
Podemos generar el código con la orden '***perform command --mavenCommand generate-sources***'.



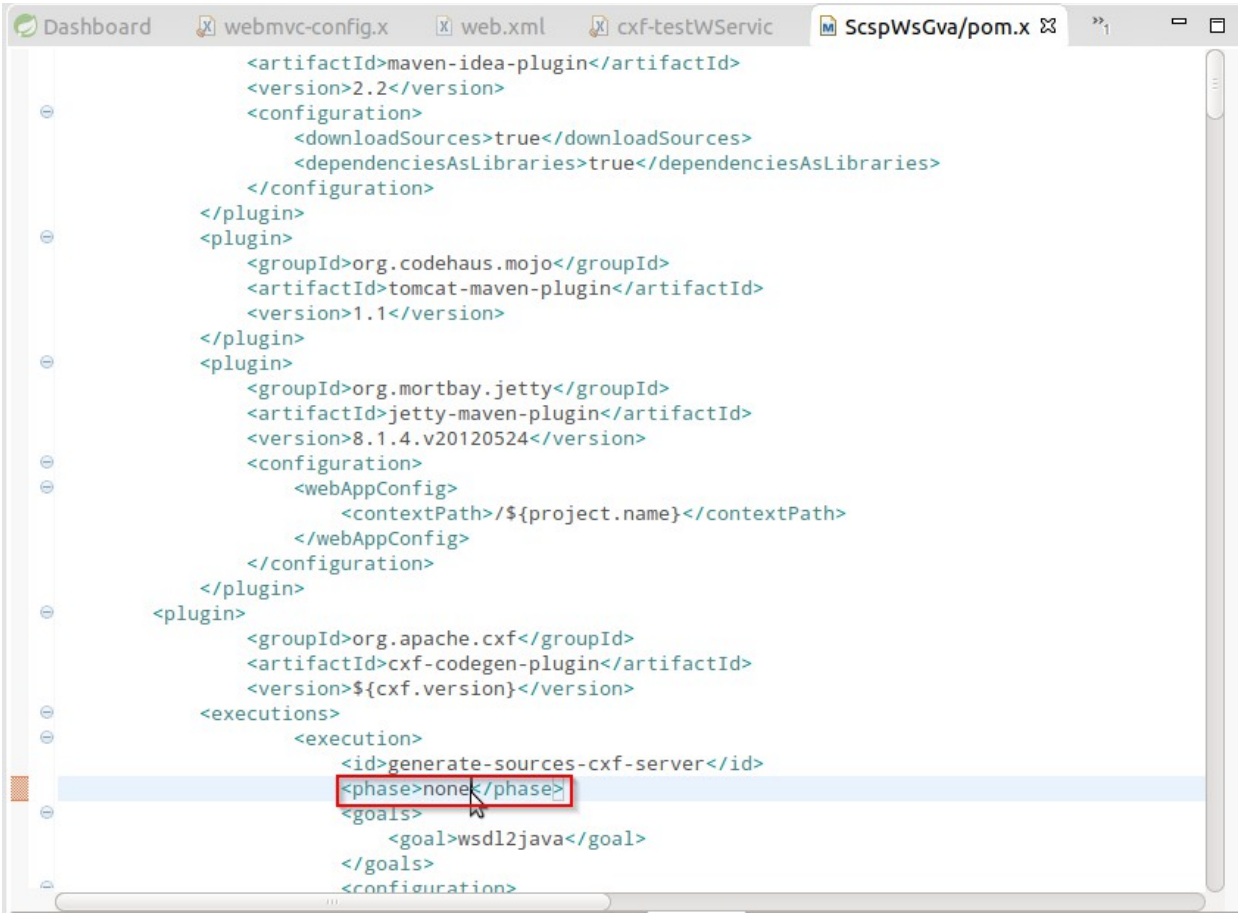
Esto dejará las clases de interés en ***target/gerated-sources-cxf-server***



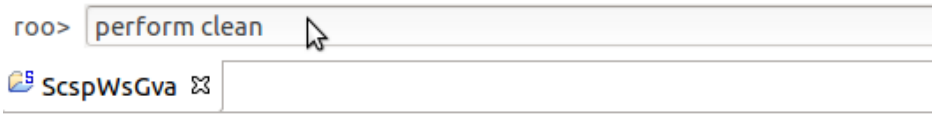
Posteriormente copiamos dichas clases a ***src/main/java***



Una vez copiados dejamos la sección ***<phase>*** del pom, de nuevo a '***none***' para que no regenere de nuevo el código.



Ejecutamos el comando '***perform clean***'

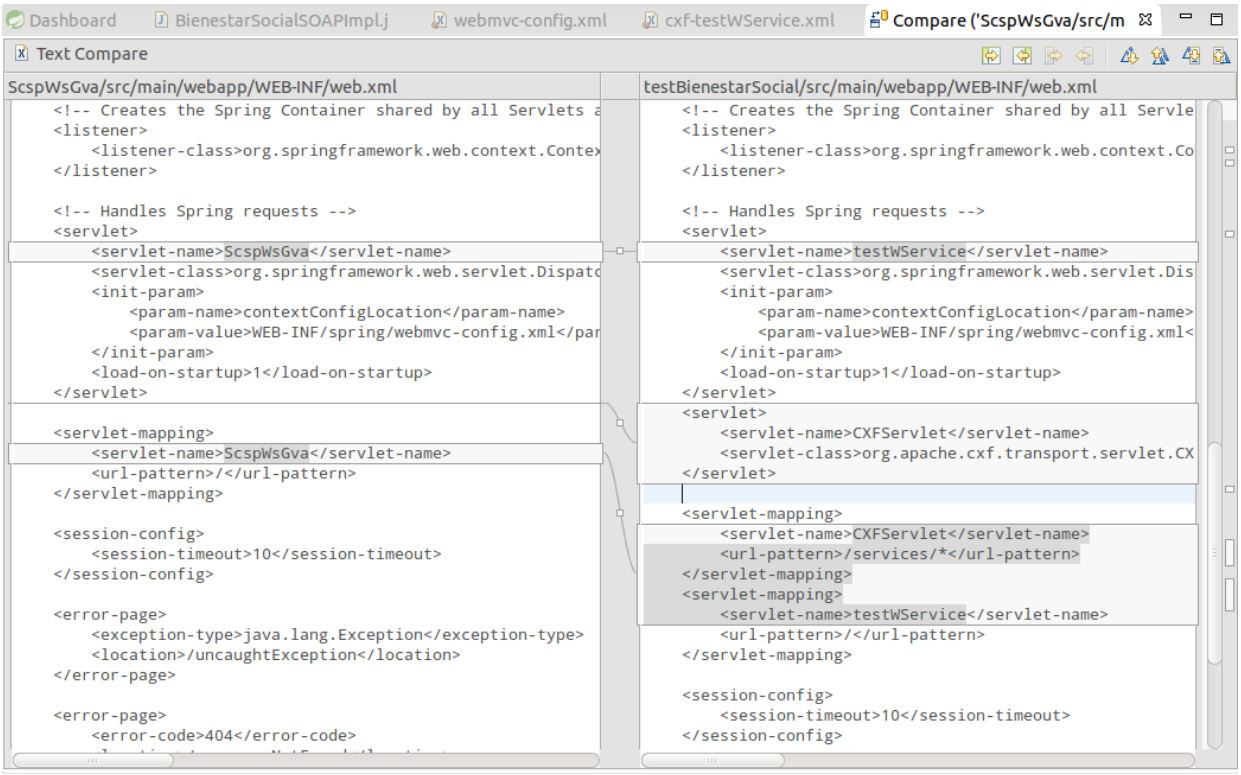


Finalmente hemos de modificar los archivos :

- **web.xml**: Incluyendo los servlets de interés:

```
<servlet>
	<servlet-name>CXFServlet</servlet-name>
	<servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>
</servlet>
<servlet-mapping>
	<servlet-name>CXFServlet</servlet-name>
	<url-pattern>/services/*</url-pattern>
</servlet-mapping>
<servlet-mapping>
	<servlet-name>testWService</servlet-name>
```

```
            <url-pattern>/</url-pattern>
        </servlet-mapping>
```



- **cxf-testService.xml:** en caso de no encontrarse en la carpeta src/main/webapp/WEB-INF, hemos de crearlo y posteriormente referenciarlo en el fichero web.xml como vemos a continuación:

  - **cxf-testService.xml**

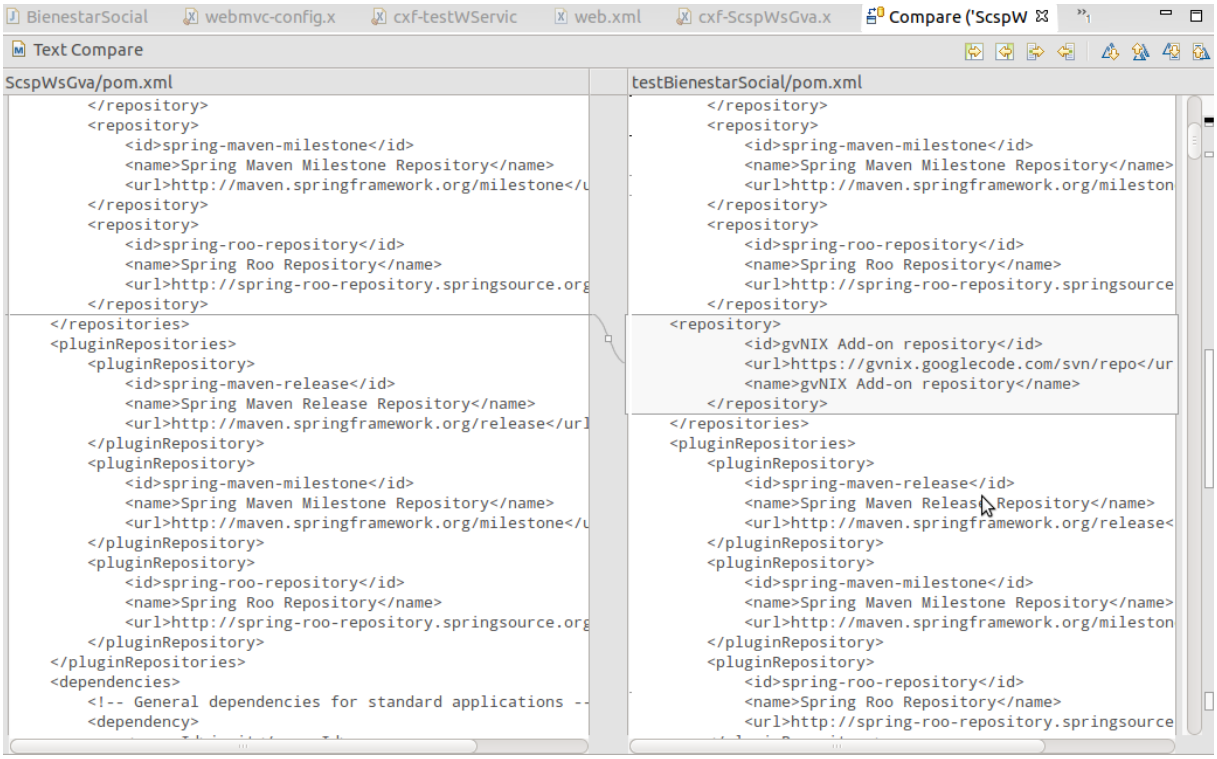
```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
       xmlns:tx="http://www.springframework.org/schema/tx"
xmlns:util="http://www.springframework.org/schema/util"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:jaxws="http://cxf.apache.org/jaxws"
       xsi:schemaLocation="
            http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
            http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util-3.0.xsd
            http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd
            http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.0.xsd
```

```
            http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.0.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="TituloFamiliaNumerosa"
                class="es.redsara.intermediacion.xml_schemas.TituloFamiliaNumerosaSOAPImpl" />

        <jaxws:endpoint id="TituloFamiliaNumerosajws"
                implementor="#TituloFamiliaNumerosa" address="/TituloFamiliaNumerosaSW">
        </jaxws:endpoint>
</beans>
```

- **web.xml**

```
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>WEB-INF/cxf-ScspWsGva.xml classpath*:META-
INF/spring/applicationContext*.xml</param-value>
    </context-param>
```

- **pom.xml:** Añadir las dependencias necesarias.



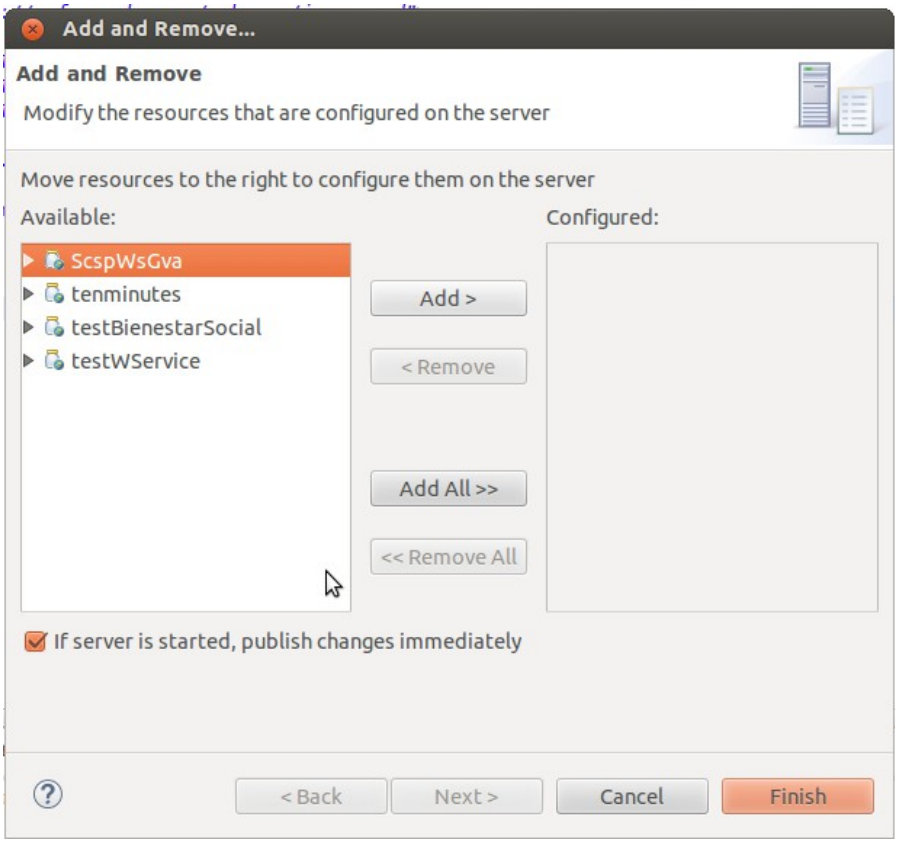
```
------------------------------------------------------------------
REPOSITORY
------------------------------------------------------------------
<repository>
            <id>gvNIX Add-on repository</id>
            <url>https://gvnix.googlecode.com/svn/repo</url>
            <name>gvNIX Add-on repository</name>
        </repository>
------------------------------------------------------------------
DEPENDENCIAS
------------------------------------------------------------------

<dependency>
            <groupId>org.gvnix</groupId>
            <artifactId>org.gvnix.service.roo.addon</artifactId>
            <version>1.2.1-RELEASE</version>
            <scope>provided</scope>
        </dependency>
    <dependency>
            <groupId>org.apache.cxf</groupId>
            <artifactId>cxf-rt-core</artifactId>
            <version>${cxf.version}</version>
        </dependency>
    <dependency>
            <groupId>org.apache.cxf</groupId>
            <artifactId>cxf-rt-bindings-soap</artifactId>
            <version>${cxf.version}</version>
        </dependency>
    <dependency>
            <groupId>org.apache.cxf</groupId>
            <artifactId>cxf-rt-databinding-jaxb</artifactId>
            <version>${cxf.version}</version>
        </dependency>
    <dependency>
            <groupId>org.apache.cxf</groupId>
            <artifactId>cxf-rt-frontend-jaxws</artifactId>
            <version>${cxf.version}</version>
        </dependency>
    <dependency>
            <groupId>org.apache.cxf</groupId>
            <artifactId>cxf-rt-transports-http</artifactId>
            <version>${cxf.version}</version>
        </dependency>
    <dependency>
            <groupId>javax.xml.bind</groupId>
            <artifactId>jaxb-api</artifactId>
            <version>2.2.7</version>
        </dependency>
    <dependency>
            <groupId>javax.xml.ws</groupId>
            <artifactId>jaxws-api</artifactId>
            <version>2.2.8</version>
        </dependency>
    <dependency>
```

```
            <groupId>org.springframework</groupId>
            <artifactId>spring-tx</artifactId>
            <version>${spring.version}</version>
        </dependency>
    <dependency>
            <groupId>org.hsqldb</groupId>
            <artifactId>hsqldb</artifactId>
            <version>2.2.9</version>
        </dependency>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>4.2.2.Final</version>
        </dependency>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-entitymanager</artifactId>
            <version>4.2.2.Final</version>
            <exclusions>
                <exclusion>
                    <groupId>cglib</groupId>
                    <artifactId>cglib</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>dom4j</groupId>
                    <artifactId>dom4j</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>org.hibernate.javax.persistence</groupId>
            <artifactId>hibernate-jpa-2.0-api</artifactId>
            <version>1.0.1.Final</version>
        </dependency>
        <dependency>
            <groupId>commons-collections</groupId>
            <artifactId>commons-collections</artifactId>
            <version>3.2.1</version>
        </dependency>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-validator</artifactId>
            <version>4.3.1.Final</version>
        </dependency>
        <dependency>
            <groupId>javax.validation</groupId>
            <artifactId>validation-api</artifactId>
            <version>1.0.0.GA</version>
        </dependency>
        <dependency>
            <groupId>javax.transaction</groupId>
            <artifactId>jta</artifactId>
            <version>1.1</version>
        </dependency>
```

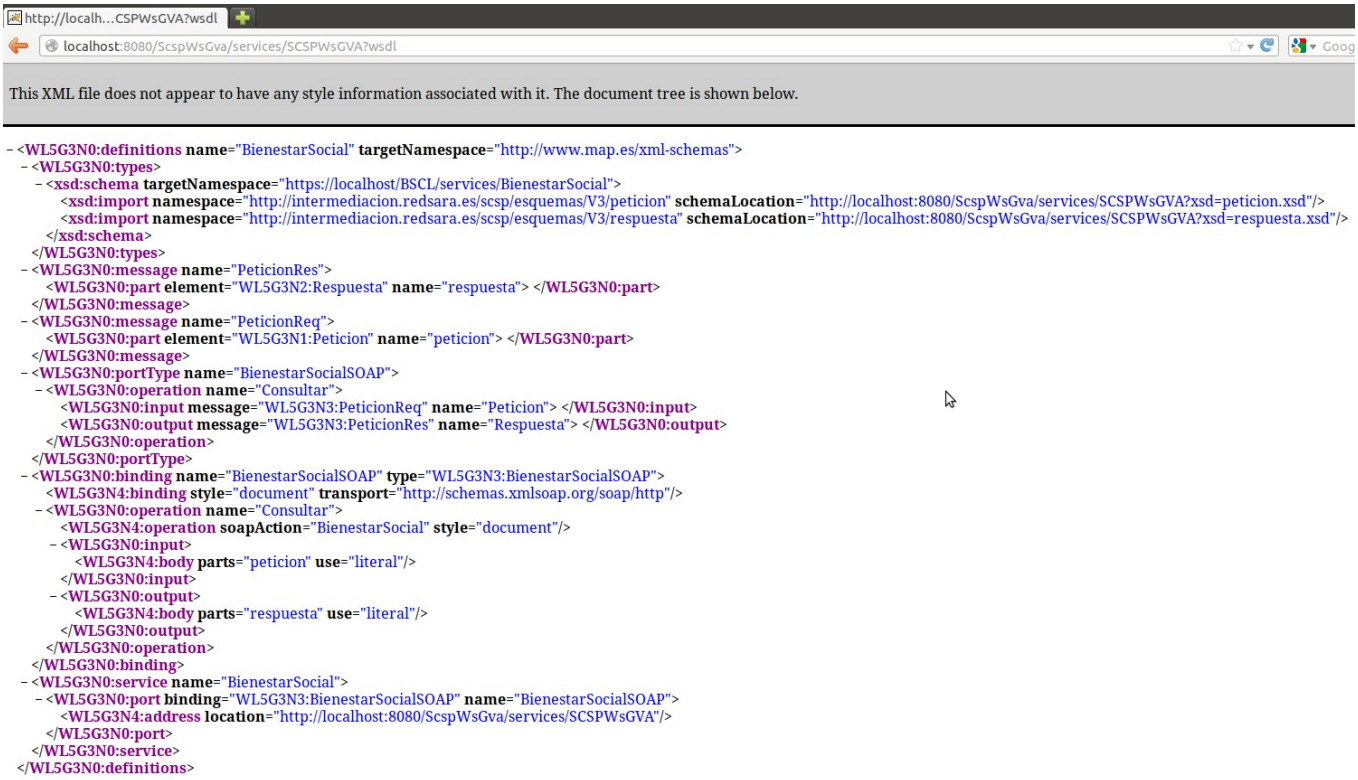
```
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-orm</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>commons-pool</groupId>
            <artifactId>commons-pool</artifactId>
            <version>1.5.6</version>
        </dependency>
        <dependency>
            <groupId>commons-dbcp</groupId>
            <artifactId>commons-dbcp</artifactId>
            <version>1.4</version>
            <exclusions>
                <exclusion>
                    <groupId>commons-logging</groupId>
                    <artifactId>commons-logging</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>xml-apis</groupId>
                    <artifactId>xml-apis</artifactId>
                </exclusion>
            </exclusions>
        </dependency>

-----------------------------------------------------------------
PLUGIN
-----------------------------------------------------------------
<plugin>
                <groupId>org.apache.cxf</groupId>
                <artifactId>cxf-java2ws-plugin</artifactId>
                <version>${cxf.version}</version>
                <dependencies>
                    <dependency>
                        <groupId>org.apache.cxf</groupId>
                        <artifactId>cxf-rt-frontend-jaxws</artifactId>
                        <version>${cxf.version}</version>
                    </dependency>
                    <dependency>
                        <groupId>org.apache.cxf</groupId>
                        <artifactId>cxf-rt-frontend-simple</artifactId>
                        <version>${cxf.version}</version>
                    </dependency>
                </dependencies>
            </plugin>
```

Una vez realizados estos pasos ya podemos comenzar a implementar los métodos que el wsdl tiene y CXF a traducido en clases java para implementar. La clase ***XXXXXSOAPImpl.java*** donde las X corresponden al nombre del servicio.

Una vez realizada la lógica que queramos, podemos desplegar la aplicación en el servidor configurado en el STS.

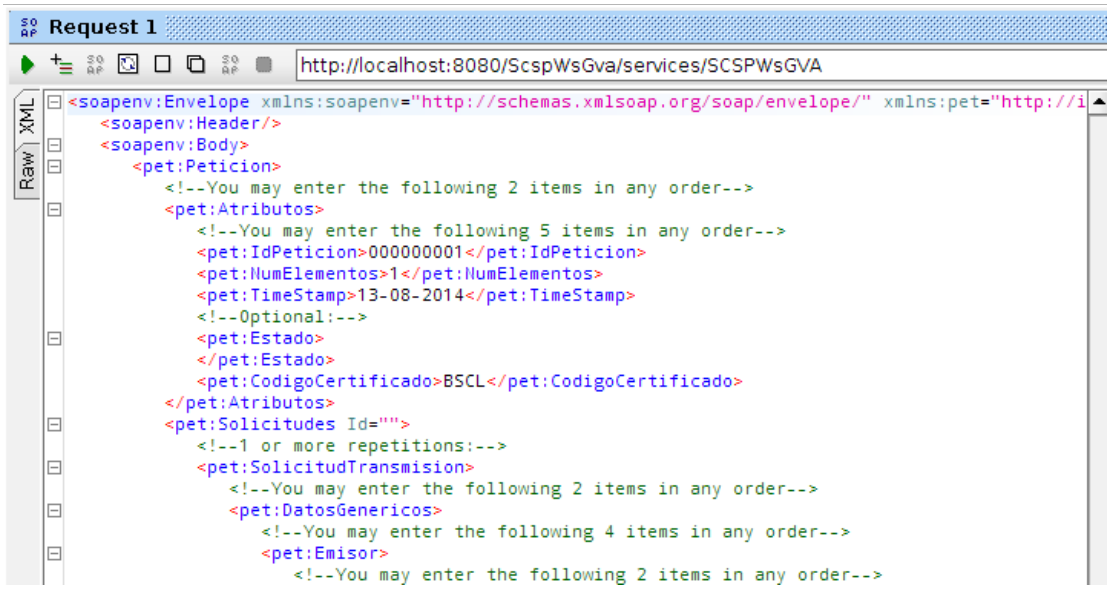


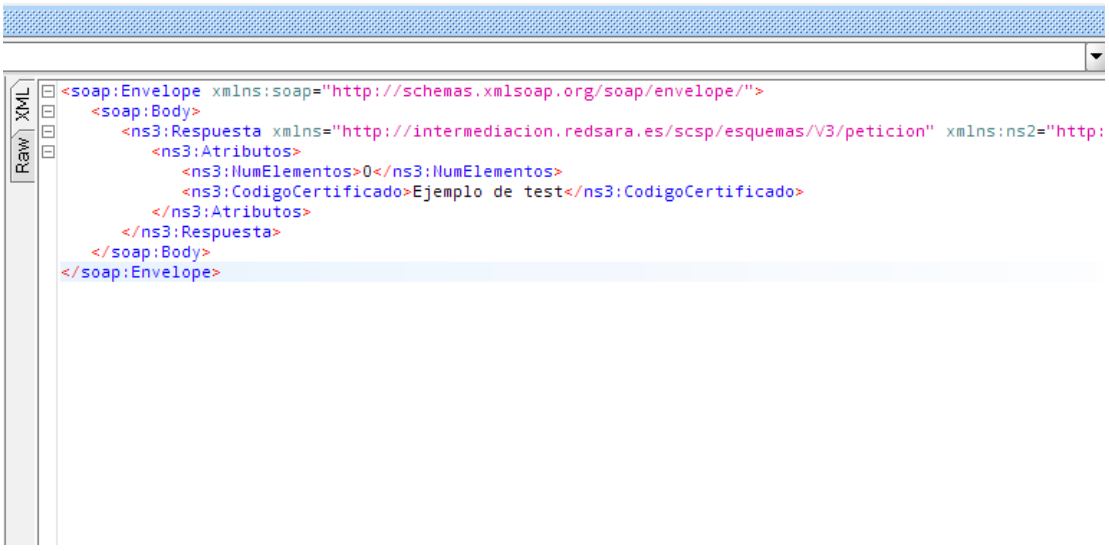
Podremos acceder al wsdl una vez desplegada la aplicación, mediante la dirección proporcionada en el ***cxf-xxxWService.xml***, donde veremos el wsdl con la estructura deseada.



SOAPUI

Podemos utilizar SoapUI como herramienta para el testeo de nuestro servicio web, una vez desplegada la aplicación gvNIX y accesible en el wsdl, podremos incluirlo en el SOAPUI para realizar peticiones de prueba y verificar la respuesta.


Request 1
http://localhost:8080/ScspWsGva/services/SCSPWsGVA
XML
Raw
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:pet="http://i
<soapenv:Header/>
<soapenv:Body>
<pet:Peticion>
<!--You may enter the following 2 items in any order-->
<pet:Atributos>
<!--You may enter the following 5 items in any order-->
<pet:IdPeticion>000000001</pet:IdPeticion>
<pet:NumElementos>1</pet:NumElementos>
<pet:TimeStamp>13-08-2014</pet:TimeStamp>
<!--Optional:-->
<pet:Estado>
</pet:Estado>
<pet:CodigoCertificado>BSCL</pet:CodigoCertificado>
</pet:Atributos>
<pet:Solicitudes Id="">
<!--1 or more repetitions:-->
<pet:SolicitudTransmision>
<!--You may enter the following 2 items in any order-->
<pet:DatosGenericos>
<!--You may enter the following 4 items in any order-->
<pet:Emisor>
<!--You may enter the following 2 items in any order-->


XML
Raw
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ns3:Respuesta xmlns="http://intermediacion.redsara.es/scsp/esquemas/V3/peticion" xmlns:ns2="http:
<ns3:Atributos>
<ns3:NumElementos>0</ns3:NumElementos>
<ns3:CodigoCertificado>Ejemplo de test</ns3:CodigoCertificado>
</ns3:Atributos>
</ns3:Respuesta>
</soap:Body>
</soap:Envelope>

4 Anexo II. Añadir WS-Security Signature

En este apartado vamos a tratar de explicar como añadir la cabecera de seguridad la respuesta del servicio con ws-security mediante un certificado.

Para ello y tomando como base el proyecto desarrollado en los apartados anteriores, deberemos realizar varias acciones sobre el mismo.

En primer caso creamos el "***properties***" donde irán todas las propiedades del certificado que vayamos a utilizar en la firma del mensaje e incluir el mismo en una ruta accesible.

```
org.apache.ws.security.crypto.merlin.keystore.file=serviceKeystore.jks
org.apache.ws.security.crypto.merlin.keystore.password=sspass
org.apache.ws.security.crypto.merlin.keystore.type=jks
org.apache.ws.security.crypto.merlin.keystore.alias=myservicekey
####
wss.serviceKeyAlias=myservicekey
wss.keypass=skpass
```

Por otro lado, debemos incluir un interceptor en la salida del mensaje para que mediante una clase referenciada en el mismo cree la cabecera arreglo a los parámetros que le pasemos. Para ello nos dirigimos al fichero de propiedades de cxf y añadimos las siguientes propiedades.

```
      <bean id="keystorePasswordCallback"
class="com.example.tutorial.ws.security.KeystorePasswordCallback" />
<jaxws:endpoint id="SCSPWsGVAjws" implementor="#SCSPWsGVA" address="/SCSPWsGVA">
      <jaxws:outInterceptors>
            <ref bean="wss4jOutConfiguration" />
      </jaxws:outInterceptors>
</jaxws:endpoint>
<bean id="wss4jOutConfiguration"
class="org.apache.cxf.ws.security.wss4j.WSS4JOutInterceptor">
      <property name="properties">
            <map>
                  <entry key="action" value="Timestamp Signature" />
                  <entry key="user" value="myservicekey" />
                  <entry key="mustUnderstand" value="false" />
                  <entry key="signaturePropFile" value="serviceKeystore.properties" />
                  <entry key="passwordCallbackClass"
value="com.example.tutorial.ws.security.KeystorePasswordCallback" />
                  <entry key="signatureAlgorithm"
value="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
            </map>
      </property>
</bean>
```

Como observamos, en la declaración del bean incluimos las propiedades de la firma de las cuales podemos destacar:

```
<entry key="action" value="Timestamp Signature" />
```

Esta declaración informa de que acciones vamos a realizar en la firma. En este caso consideramos que añadir la firma y el timestamp es lo correcto.

```
<entry key="user" value="myservicekey" />
```

Alias del usuario del keystore configurado que vamos a utilizar.

```
<entry key="mustUnderstand" value="false" />
```

Propiedad que informa de si incluiremos o no el parámetro "***mustUnderstand***" en la cabecera.

```
<entry key="signaturePropFile" value="serviceKeystore.properties" />
```

Enlazamos el ***properties*** donde va referenciado el jks en este caso.

```
<entry key="passwordCallbackClass"
value="com.example.tutorial.ws.security.KeystorePasswordCallback" />
```

En este momento, indicamos la clase que va a manejar y realizar las acciones sobre el mensaje. Posteriormente describimos cual va a ser la implementación de la misma.

Debemos crear la clase manejadora "***Handler***" que actuará sobre el menaje, esta clase, extiende a '***CallbackHandler***' y en su constructor debemos hacer referencia al "***.properties***" creado anteriormente e indicarle los parámetros de alias y password con los cuales firmamos.

```
package com.example.tutorial.ws.security;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.ResourceBundle;

import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.UnsupportedCallbackException;

import org.apache.ws.security.WSPasswordCallback;

public class KeystorePasswordCallback implements CallbackHandler {

   private Map<String, String> passwords =
      new HashMap<String, String>();

   public KeystorePasswordCallback() {
         ResourceBundle rb = ResourceBundle.getBundle("serviceKeystore");
```

```
            passwords.put(rb.getString("wss.serviceKeyAlias"),rb.getString("wss.keypass"));
    }

            public void handle(Callback[] callbacks) throws IOException, UnsupportedCallbackException {
        for (int i = 0; i < callbacks.length; i++) {
            WSPasswordCallback pc = (WSPasswordCallback)callbacks[i];

            String pass = passwords.get(pc.getIdentifier());
            if (pass != null) {
                pc.setPassword(pass);
                return;
            }
        }
    }

}
```

Una vez hemos realizado estos pasos, podremos ejecutar el proyecto y comprobar mediante las herramientas de consumo que dispongamos, que el servicio creado con gvNIX está firmando las respuestas.

Encriptado de partes

Muchos de los servicios que se implementan realizan transferencias de datos sensibles, por ello, es interesante contemplar la posibilidad de encriptar parte de los mensajes, a cualquier nivel.

En este apartado vamos a tratar un ejemplo básico de como encriptar un campo del mensaje. Todas las modificaciones necesarias para ello se realizan en el fichero de configuración de cxf.

Tomamos la clase interceptor antes realizada y procedemos a ampliar las propiedades descritas dentro del bean.

```
<entry key="action" value="Timestamp Signature Encrypt"/>
```

Realizamos cambios en la action, añadiendo el valor "***Encrypt***"

```
<entry key="encryptionPropFile" value="serviceKeystore.properties"/>
<entry key="encryptionUser" value="myservicekey"/>
```

Declaramos el fichero y usuario con el que vamos a encriptar.

```
<entry key="encryptionParts" value="{Content}{http://intermediacion.redsara.es/scsp/esquemas/V3/respuesta}CodigoCertificado"/>
<entry key="encryptionSymAlgorithm" value="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
<entry key="encryptionKeyTransportAlgorithm" value="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
```

Y finamente las partes que vamos a encriptar, teniendo en cuenta que se puede encriptar a nivel de contenido (Content) o elemento (Element), así como el algoritmo a utilizar. Recordamos que se encripta la primera entrada de la lista que se identifica con el campo y pertenece al namespace. Hay que tener en cuenta que la propiedad y el nombre son keysensitive.

5 Anexo III.Trazabilidad de los servicios

Con el fin de realizar una correcta implementación del servicio, siguiendo las buenas practicas propuestas por la PAI en el documento “Desarrollo y consumo de servicios web – Buenas Practicas”, incluimos este punto donde vamos a realizar un control de la cabecera de mensaje SOAP necesaria para la trazabilidad del mensaje en la plataforma.

> *“Este tag se compone del número de serie del certificado utilizado para firmar la petición, el código de procedimiento administrativo y el instante de la petición codificado como año,mes,dia,hora,minuto,segundo. Estos tres componentes del tag de trazabilidad se han de separar por un carácter ‘-‘. Ejemplo:*
>
> *<id_trazabilidad xmlns="uri:es.i15d.interoperabilidad">68254ed222d65eeb-9999999999999999999-20140130184955</id_trazabilidad>”*

Por tanto, para ello debemos crear un interceptor en el cual indicamos la phase de su actuación y las acciones a realizar con el mensaje de entrada.

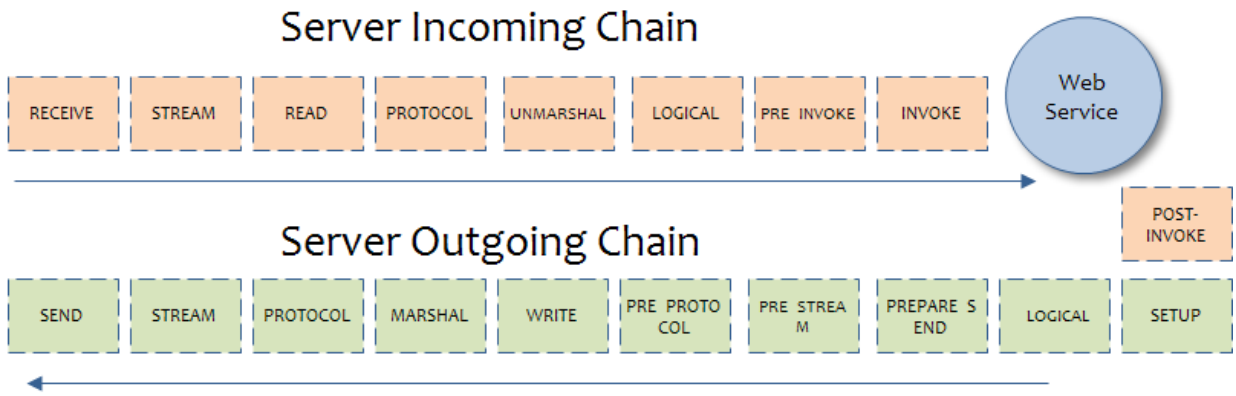
```
public class CustomSoapHeaderInInterceptor extends AbstractSoapInterceptor {
        public CustomSoapHeaderInInterceptor() {
                super(Phase.PRE_INVOKE);
        }

        @Override
        public void handleMessage(SoapMessage message) throws Fault {
                List<Header> list = message.getHeaders();
                for (Header header : list) {
                        QName qname = (QName) header.getName();
                        if(qname.getLocalPart().equals("id_trazabilidad")){
                                if (header.getObject() instanceof Element) {
                                        Element elem = (Element) header.getObject();
                                        String id_trazabilidad = elem.getFirstChild().getTextContent().toString();
                                        String[] partes_traza = id_trazabilidad.split("-");
                                        System.out.println(header.getName() + " = " +
elem.getFirstChild().getTextContent().toString());
                                        System.out.println("Certificado = " + partes_traza[0]);
                                        System.out.println("Procedimiento = " + partes_traza[1]);
                                        System.out.println("Timestamp = " + partes_traza[2]);
                                }
                        } else {
                                //Acciones si no contiene el id_trazabilidad
```

```
                    }

                }
            }
}
```

Posteriormente, debemos declarar correctamente el interceptor creado como interceptor de entrada en el fichero de configuración de cxf.

```
<jaxws:inInterceptors>
        <ref bean="headerInterceptor"/>
</jaxws:inInterceptors>

<bean                                                    id="headerInterceptor"
class="com.example.tutorial.ws.security.CustomSoapHeaderInInterceptor" />
```

Como vemos en el gráfico posterior, son varias las phases donde podemos situar dicho interceptor. En el ejemplo propuesto se sitúa en la phase de ***PRE_INVOKE***.



Este interceptor se encargará de recuperar las cabeceras SOAP y comprobar que existe. Además en el ejemplo se propone una pequeña sección de código donde se trata de separar el tag por componentes (certificado,procedimiento,timestamp).

6 Anexo IV. Añadir soapfaults tipo SCSP

En este anexo vamos a realizar las acciones correspondientes para añadir soapfaults al proyecto. Los pasos a seguir para resolver este caso son;

- Modificar el wsdl del servicio (entendiendo que disponemos del ***xsd*** que define los "***detail***" del **soapfault**)
- Regenerar las clases del servicio web, mediante el cambio en el ***pom*** y la instrucción antes utilizada.
- Programar la lógica en la implementación del servicio para lanzar el ***soapfault***.

Como primer paso, añadimos lo necesario al ***wsdl*** para que contemple los ***faults*** personalizados.

```
<wsdl:definitions
xmlns:ns3="http://intermediacion.redsara.es/scsp/esquemas/V3/soapfaultatributos"
>
```

Añadimos el namespace en la etiqueta "***definitions***", para ser utilizado posteriormente.

```
  <wsdl:types>
    <xsd:schema targetNamespace="http://intermediacion.redsara.es/xml-schemas">
            ...
            <xsd:import
namespace="http://intermediacion.redsara.es/scsp/esquemas/V3/soapfaultatributos"
schemaLocation="soapfaultatributos.xsd"/>
    </xsd:schema>
 </wsdl:types>
```

Dentro de los tipos importamos el ***xsd*** correspondiente al ***soapfault***, en este caso llamado soapfaultatributos.

```
<wsdl:message name="FaultRes">
    <wsdl:part element="ns3:Atributos" name="error"/>
  </wsdl:message>
```

Añadimos un part con el elemento Atributos que figura en el ***xsd*** y lo nombramos como error.

```
 <wsdl:portType name="TituloFamiliaNumerosaSOAP">
    <wsdl:operation name="Consultar">
      ...
        <wsdl:fault message="impl:FaultRes" name="error" />
    </wsdl:operation>
  </wsdl:portType>
```

En la interfaz portType añadimos una nuevo tipo para los ***fault***.

```
<wsdl:binding                                    name="TituloFamiliaNumerosaSOAP"
type="impl:TituloFamiliaNumerosaSOAP">
                                        <soap:binding          style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="Consultar">
      <soap:operation soapAction="TituloFamiliaNumerosa" style="document"/>
      ...
        <wsdl:fault name="error">
          <soap:fault name="error" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
```

En la implementación de la operación añadimos el mismo tipo de recurso ***fault*** definido como "***literal***".

Con esto el ***wsdl*** esta listo para hacer la operación ***wsdl2java*** de ***cxf*** que se realiza en nuestro proyecto gvNIX de la misma forma que hemos visto en punto 3.2.

Recordamos que debemos modificar la fase del pom relativa a la generación con cxf a ***generate-sources*** y utilizar la expresión '***perform command --mavenCommand generate-sources***' en la consola.

Una vez realizada la operación debemos copiar los ficheros generados a '***src/main/java***'.

La utilización de esta nueva funciona se realiza muy fácilmente con esta parte de código:

```
public es.redsara.intermediacion.scsp.esquemas.v3.respuesta.Respuesta
consultar(es.redsara.intermediacion.scsp.esquemas.v3.peticion.Peticion peticion) throws
FaultRes    {
             try {
             if(peticion.getAtributos().getIdPeticion().toString().equals("0001")){
                          throw new FaultRes("Error");
                    }

                    return _return;
             } catch (FaultRes ex) {

      es.redsara.intermediacion.scsp.esquemas.v3.soapfaultatributos.Atributos _atr =
new Atributos();
                    _atr.setIdPeticion("50");
                    throw new FaultRes("Error",_atr);
             } catch (java.lang.Exception ex) {
                    ex.printStackTrace();
```

```
                throw new RuntimeException(ex);
            }
        }
```

Donde lanzamos la excepción del tipo generado por cxf a raíz del ***wsdl*** y completamos el esquema definido para pasárselo como parámetro.