GENERALITAT VALENCIANA

Conselleria d'Hisenda i Model Econòmic

Integración de WS en aplicación Java

E3S 3.0

Versión 1.0

14 de agosto de 2023

Control del documento

Título	Integración de WS en aplicación Java
Nombre archivo con ubicación	C:\Proyectos\ADCR_Documentacion\09_doc_adicional\Integracion_WS_en_aplicacion_Java.odt
Tipo	☐ Documento de trabajo ☐ Documento de referencia
Clasificación	☐ Público ☐ Interno ☐ Restringido ☐ Confidencial
Estado	☐ Borrador ☐ Aprobado ☐ Obsoleto

Control de cambios

Versión	Estado	Responsable	Organismo	Descripción del cambio	Fecha
1.0	Borrador	Alejandro Santonja	SOPRA-Tecnocom	Versión Inicial	08/07/2021
1.1	Borrador	Nerea Jordan	SOPRA STERIA-ODEC-PENTEC	Indicaciones para inserción de Certificado	11/08/2023

Índice de contenido

1. OBJETIVO DEL DOCUMENTO

El objetivo del documento es indicar el procedimiento a seguir para integrar los servicios de E3S en una aplicación Java. El procedimiento para crear integrar todos los servicios es el mismo, por lo que vamos a realizar estos pasos con uno de ellos:

https://residuos.gva.es/e3s/queryEnvironmentalMasterData.svc?wsdl

2. PASOS A REALIZAR

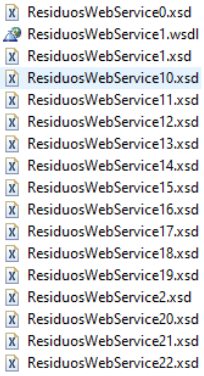
Lo primero que hay que comprobar es si accedemos al servicio web. Simplemente con introducir la url (https://residuos.gva.es/e3s/queryEnvironmentalMasterData.svc?wsdl) deberá salirnos una página similar a la siguiente:

Esto significa que tenemos acceso, y que por lo tanto podemos proceder con los siguientes pasos.

La url indicada nos dice el xml a partir del cual se realiza un import. En este caso, como se puede ver en dicha página, la url es http://residuos.gva.es/e3s/queryEnvironmentalMasterData.svc?wsdl=wsdl0

Este import es al que accederemos y copiaremos el xml que nos muestra y generaremos en nuestro local el fichero wsdl. Como podemos observar dentro de este xml existen varios import a los xsd, estos import apuntan a urls externas de nuestro local, tendremos que acceder a ellas a través del navegador y crear los diferentes ficheros xsd en nuestro local ya que CXF no funciona correctamente con urls externas.

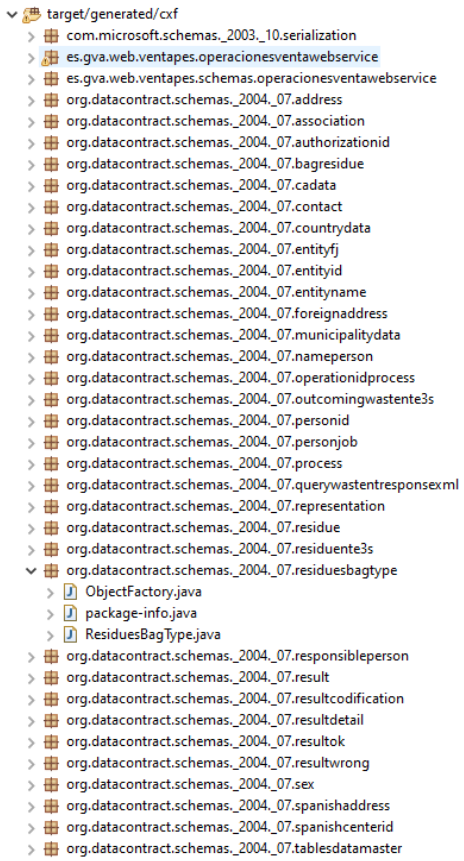
Primero crearemos el WSDL y el XSD a partir de las urls indicadas en cada servicio.



Con la ayuda de la librería CXF, se auto generaran las clases necesarias para el request y response a parte del PortType donde se implantan las llamadas al WS.

```
 */
@WebService(targetNamespace = "http://tempuri.org/", name = "IwsSendWasteNT")
@XmlSeeAlso({org.datacontract.schemas._2004._07.address.ObjectFactory.class, org.datacontract.schemas._2004._07.result.ObjectFactory.class, org.datacontract.schemas._2004._07.entityid.Object
@SOAPBinding(parameterStyle = SOAPBinding.ParameterStyle.BARE)
public interface IwsSendWasteNT {

    @WebMethod(action = "http://tempuri.org/IwsSendWasteNT/sendWasteNT")
    @Action(input = "http://tempuri.org/IwsSendWasteNT/sendWasteNT", output = "http://tempuri.org/IwsSendWasteNT/sendWasteNTResponse")
    @WebResult(name = "OutWasteNTResponseXML", targetNamespace = "http://tempuri.org/", partName = "parameters")
    public OutWasteNTResponseXML sendWasteNT(
        @WebParam(partName = "parameters", name = "sendWasteNTXML", targetNamespace = "http://tempuri.org/")
        SendWasteNTXML parameters
    );
}
```



Una vez comprobado que se ha generado correctamente estas clases, procederemos a indicar los parámetros que deseemos para preparar el request.

```
org.datacontract.schemas._2004._07.wasteproducercentere3s.ObjectFactory factoryWasteProducer = new org.datacontract.schemas.
org.datacontract.schemas._2004._07.authorizationid.ObjectFactory factoryAuthorizationId = new org.datacontract.schemas._2004
org.datacontract.schemas._2004._07.residuente3s.ObjectFactory factoryresiduente = new org.datacontract.schemas._2004._07.res
org.datacontract.schemas._2004._07.process.ObjectFactory factoryproccess = new org.datacontract.schemas._2004._07.process.Ob
org.datacontract.schemas._2004._07.bagresidue.ObjectFactory factorybagresidue = new org.datacontract.schemas._2004._07.bagre
org.datacontract.schemas._2004._07.tablesdatamastere3s.ObjectFactory factorytables = new org.datacontract.schemas._2004._07.
org.datacontract.schemas._2004._07.operationidprocess.ObjectFactory factoryoperation = new org.datacontract.schemas._2004._0
org.datacontract.schemas._2004._07.wastetransferoperatorcentere3s.ObjectFactory factorytransfer = new org.datacontract.schem
SendWasteNTXML parameters = factory.createSendWasteNTXML();

WasteNTE3S wasteNTE3S = new WasteNTE3S();

WasteManagerCenterE3S wasteManagerCenterE3S = new WasteManagerCenterE3S();


//NTAdresseeData
//AuthorizationId
AuthorizationId authorizationId = new AuthorizationId();
authorizationId.setAuthorizationIdEffects(factoryAuthorizationId.createAuthorizationIdAuthorizationIdEffects("true"));
authorizationId.setAuthorizationIdFree(factoryAuthorizationId.createAuthorizationIdAuthorizationIdFree("135/G02/RP/CV"));
authorizationId.setAuthorizationIdNumber(factoryAuthorizationId.createAuthorizationIdAuthorizationIdNumber("true"));
wasteManagerCenterE3S.setAuthorizationId(factoryAuthorizationId.createAuthorizationId(authorizationId));
wasteManagerCenterE3S.setCenterCode(factoryWasteManager.createWasteManagerCenterE3SCenterCode("4600031193"));
//TO DO
wasteManagerCenterE3S.setCenterOperator(factoryWasteManager.createWasteManagerCenterE3SCenterOperator(value));

wasteManagerCenterE3S.setManagerTypeCode(factoryWasteManager.createWasteManagerCenterE3SManagerTypeCode("G02"));
wasteManagerCenterE3S.setMunicipalityCode(factoryWasteManager.createWasteManagerCenterE3SMunicipalityCode("462508"));
wasteManagerCenterE3S.setName(factoryWasteManager.createWasteManagerCenterE3SName("Nerea"));
wasteManagerCenterE3S.setNationalID(factoryWasteManager.createWasteManagerCenterE3SNationalID("25414962P"));
wasteManagerCenterE3S.setSexCode(factoryWasteManager.createWasteManagerCenterE3SSexCode("9"));
```

En este caso nos hemos ayudado de las clases ObjectFactory que se han generado automáticamente para la creación del request.

Cuando hayamos creado correctamente el request, procederemos a enviarlo al WS, para esto se invocará al método creado en el PorType.

Este método enviará al WS el request y nos proporcionará un response con el resultado del WS, este response lo podremos gestionar como deseemos.

```
OutWasteNTResponseXML save = residuosSoapService.sendWaste(parameters);

System.out.println(save.getXmlSendWasteNTResponseElement().getValue().toString());
```

Indicar que hay que crear el bean en el configuration para establecer la conexión previa con el WS.

```
@Bean
public ResiduosSoapService residuosSoapClient() {
  return new ResiduosSoapServiceImpl();
}
```

Añadimos en la configuración de la llamada la firma con el certificado que nos han dado autorización.

Este certificado se debe de adjuntar El certificado se debe de incorporar como Signature con:

- Key Identifier Type: Binary Segurity Token
- Signature Algorithm: http://www.w3.org/2000/09/xmldsig#rsa-sha1
- Signature Canonicalization: http://www.w3.org/2001/10xml-exc-c14n#
- Digest Algorithm: https://www.w3.org/2000/09/xmldsig#sha1

```
public ResiduosSoapServiceImpl() {
    Long tiempoEspera = null;
    // Crea y configura la factoría de generación del servicio
    JaxWsProxyFactoryBean factoria = new JaxWsProxyFactoryBean();
    factoria.setServiceClass(IwsSendWasteNT.class);
    factoria.setAddress("https://residuos.gva.es/e3s/sendWasteNT.svc?wsdl");

     // Registra features: mostrar mensaje en logs e incluir id traza
    factoria.setFeatures(
        Arrays.asList(new LoggingFeature(), new TraceeCxfFeature()));

    Map<String, Object> propsPeticion = new HashMap<>();
    propsPeticion.put(WSHandlerConstants.ACTION, WSHandlerConstants.SIGNATURE);
    propsPeticion.put(WSHandlerConstants.SIGNATURE_USER, "firma");
    propsPeticion.put(WSHandlerConstants.PW_CALLBACK_REF, new WsPasswordHandler("xxxxx"));
    propsPeticion.put(WSHandlerConstants.SIG_PROP_FILE, "xxxxxxxxx");
    propsPeticion.put(WSHandlerConstants.SIG_KEY_ID, "DirectReference");
    propsPeticion.put(WSHandlerConstants.IS_BSP_COMPLIANT, "false");
    propsPeticion.put(WSHandlerConstants.MUST_UNDERSTAND, "0");
    factoria.setOutInterceptors(
            Arrays.asList(new WSS4JOutInterceptor(propsPeticion)));


    // Crea el servicio a partir de su factoría
    IwsSendWasteNT servicioBase = (IwsSendWasteNT) factoria.create();

    // Conducto de comunicación del cliente para personalizar la petición
    Client cliente = ClientProxy.getClient(servicioBase);
```

Y el método que hace la llamada al WS que tenemos en el PortType.

```
    private final IwsSendWasteNT  servicio;

        @Override
        public OutWasteNTResponseXML sendWaste(SendWasteNTXML  parameters) {

                return servicio.sendWasteNT(parameters);

        }
```

Hay que tener en cuenta, que CXF automáticamente escanea la carpeta donde tenemos ubicados estos ficheros en el ejemplo, pero pudiera dar algún problema o incluso tenerlo en otra ruta para ello, tendremos que añadir estas líneas en el pom.xml

<plugin>

```
                <groupId>org.apache.cxf</groupId>
                <artifactId>cxf-codegen-plugin</artifactId>
                <executions>
                    <execution>
                        <id>generate-sources</id>
                        <configuration>
                            <wsdlOptions>
                                <wsdlOption>
                                    <!-- Contratos (WSDLs) de los servicios -->
                                    <wsdl>${project.basedir}/src/main/resources/wsdl/ResiduosWebServi -
ce1.wsdl</wsdl>
                                </wsdlOption>
                            </wsdlOptions>
                        </configuration>
                        <goals>
                            <goal>wsdl2java</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
```

Indicando la ruta específica y el wsdl.