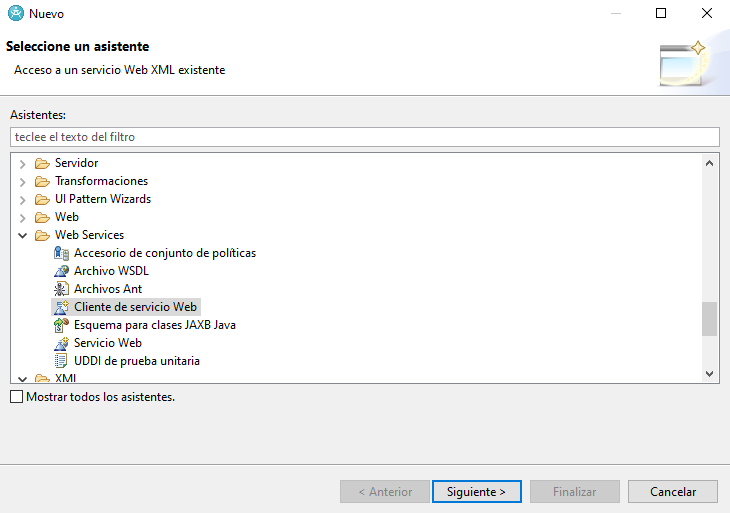
# GENERACIÓN CLIENTE SERVICIO WEB SGDA

En esta guía veremos cómo crear un las clases que implementan el cliente de un servicio Web a partir de su WSDL (Web Service Definition Language).

Para automatizar la creación de las clases del servicio, se utilizará Eclipse RSA, aunque también se podría realizar mediante el comando java ‘wsimport’, como se verá en el apartado correspondiente.

## 1.-Crear Nuevo Proyecto Cliente Servicio Web

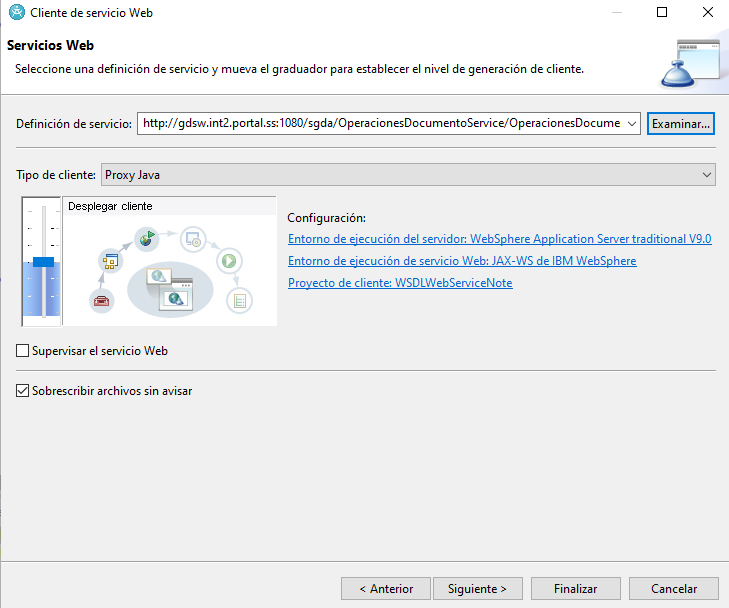
En RSA*, ‘Archivo > Nuevo > Otras…*’, seleccionar bajo la opción ‘Web Services’ la opción ‘Cliente de servicio Web’ .



## 2.-Seleccionar el WSDL del servicio del que generar el cliente

En este caso, queremos generar un cliente para invocar el servicio OperacionesDocumentoService de SGDA, por lo que lo seleccionamos indicando directamente su URL.

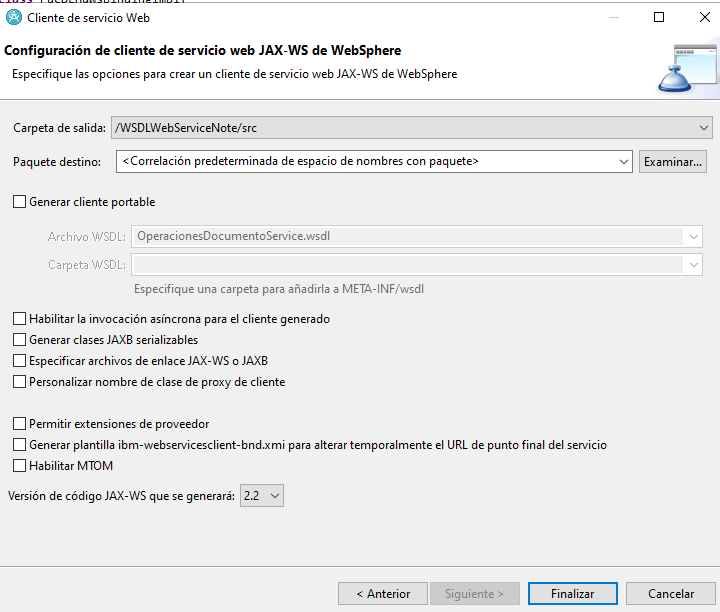
<http://gdsw.des.portal.ss:1080/sgda/OperacionesDocumentoService/OperacionesDocumentoService.wsdl>



Pulsar el botón ‘Siguiente’.

## 3.-Opciones de personalización del cliente

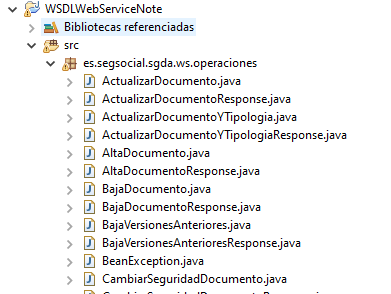
Se eliminan todas las opciones de personalización, ya que en este caso no aplica ninguna de ellas.



Por último, se pulsa el botón ‘Finalizar’.

## 3.-Comprobación Clases Generadas

Una vez RSA finaliza el proceso de generación de las clases, es importante comprobar que se han creado en el proyecto destino las clases autogeneradas que implementan el cliente.



## 3.-Probar Cliente con main standalone

Para probar que el cliente se ha generado correctamente y que accede al servicio Web, basta con generar una clase de pruebas como la indicada a continuación.

**public** **class** AccesoASGDAFichero {

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

// endpoint del servicio (estará en una JNDI)

String endpointWS = "http://gdsw.int2.portal.ss:1080/sgda/OperacionesDocumentoService";

URL url = **new** URL(endpointWS + "?wsdl");

QName qname = **new** QName("http://operaciones.ws.sgda.segsocial.es/", "OperacionesDocumentoService");

OperacionesDocumentoService servicio = **new** OperacionesDocumentoService(url, qname);

OperacionesDocumento port = servicio.getOperacionesDocumentoPort();

// Parametros de conexion a SGDA

Conexion conexion = **new** Conexion();

conexion.setUsuarioVirtual("XXXXXXXX"); // Usuario Virtual (estará en una JNDI)

conexion.setRochade("NOTE");

conexion.setSilconUsuario("99GU9999");

// Identificador del documento (docname)

Identificador documento = **new** Identificador();

documento.setId("SGDA\_XXXXXXXX"); // Identificador del documento (docname)

**try** {

// Llamada a SGDA

Fichero fich = port.obtenerFicheroDocumento(conexion, documento);

System.***out***.println("Nombre Fichero:" + fich.getNombreFichero());

// Bytes del fichero

**byte**[] fichero = fich.getFichero();

} **catch** (ConexionException\_Exception e) {

e.printStackTrace();

} **catch** (SGDAException e) {

e.printStackTrace();

}

}

}

En esta clase, se llama a la operación de obtener FicheroFocumento, que accede al contenido físico del fichero almacenado en SGDA / Documentum y lo recupera.

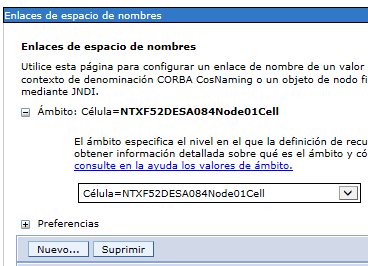
Hay que notar que esto es una clase de pruebas, en el código real de la aplicación, tanto el endpoint como el usuario virtual deberán sacarse de variables JNDI almacenadas en Websphere, ya que tanto el endpoint como el usuario virtual de SGDA varían entre entornos.

# CREACIÓN DE VARIABLES JNDI

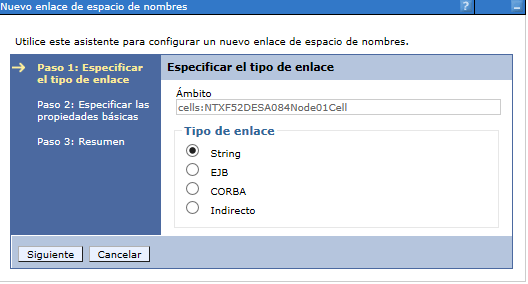
Como el usuario virtual de acceso a SGDA y el endpoint del servicio varían entre entornos, es necesario crear unas variables JNDI para recuperar sus valores.

## Variable JNDI Usuario Virtual – wdeh/virtualSGDA

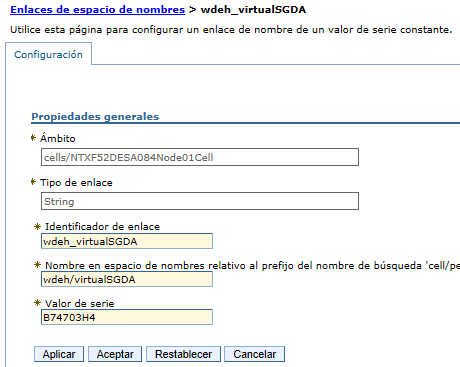
Desde la consola administrativa de Websphere: ‘Entorno > Denominación > Enlaces de espacios de nombres’:



Añadir la nueva variable JNDI de tipo String:



Asignar la nueva JNDI



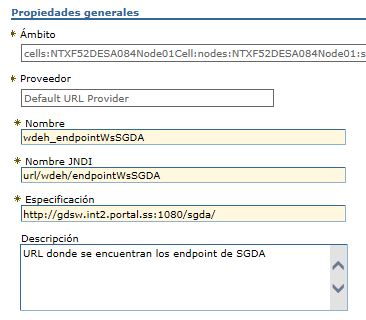
**Identificador de Enlace:** wdeh\_virtualSGDA

**Nombre en espacio de nombres:** wdeh/virtualSGDA

**Valor de serie:** B74703H4 (en DG) / C55912M8 (en Integración)

## Variable JNDI URL Endpoint SGDA – url/wdeh/endpointWsSGDA

Desde la consola administrativa de Websphere: ‘Recursos > URL > URL’:

****

**Identificador de Enlace:** wdeh\_endpointWsSGDA

**Nombre en espacio de nombres:** url/wdeh/ endpointWsSGDA

**Valor de serie:**

**DG**: <http://gdsw.des.portal.ss:1080/sgda/>

**INT**:<http://gdsw.int2.portal.ss:1080/sgda/>

# ALTERNATIVA - GENERACIÓN CLIENTES JAX-WS – SGDA CON WSIMPORT

Otra alternativa a la generación de un cliente de servicios Web con JAX-WS que es también muy sencilla, es invocar el comando **wsimport**, disponible en cualquier JDK 1.6.X de Java y en adelante.

En este ejemplo, vamos a utilizar la JDK del Websphere, que en nuestro equipo la tenemos situada en la ruta:

C:\IBM\WebSphere\AppServer\java\bin

Dentro de la carpeta bin, está el ejecutable wsimport.exe. Esta herramienta es la encargada de generar las clases necesarios que conforman un cliente de Servicios Web partiendo de los WSDL de los servicios que queremos llamar.

Situándonos en el directorio **bin**, procedemos a ejecutar el **wsimport**:

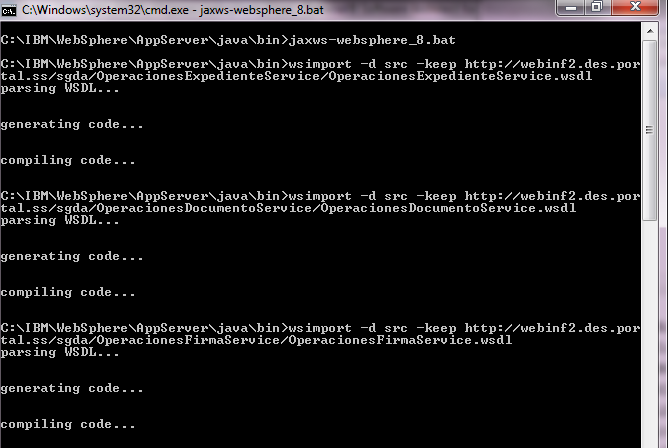
wsimport -d src -keep <http://gdsw.des.portal.ss:1080/sgda/OperacionesDocumentoService/OperacionesDocumentoService.wsdl>

donde:

**-d :** Indica el directorio donde se generarán las clases autogeneradas del cliente del servicio Web. En este caso, lo almacenaremos en una capeta ‘src’.

**-keep :** Indica que queremos que se mantengan los ficheros generados.

**wsdl:** Es necesario indicar la ruta del fichero wsdl de los servicios Web.



Una vez ejecutado el comando, en el directorio ‘src’ se habrán generado las clases necesarias para invocar al servicio web. Basta con importar las clases a nuestro proyecto RSA para poder utilizarlo.

# UTILIZACIÓN CLASES AUTOGENERADAS

Por ejemplo, para realizar la consulta del fichero de un documento almacenado en SGDA, podría hacerse uso de las clases autogeneradas de la siguiente manera:

1.-Recuperación del puerto del servicio web de SGDA (objeto OperacionesDocumento) a través de su dirección.

2.-Crear Parámetros de conexión indicando el usuario virtual (objeto Conexion) de la aplicación.

3.-Realizar la llamada a la operación correspondiente con los parámetros indicados en la petición. Por ejemplo, para la consulta del fichero del documento, sólo es necesario utilizar la clase autogenerada ‘Identificador’ (port.obtenerFicheroDocumento(conexión, idDoc)), donde se le pasa el identificador SGDA correspondiente (docName).

/\*\*

\* Método que recibe el identificador de un documento en SGDA (docName - SGDA\_XXXXXX)

\* y devuelve el fichero físico del mismo

\* **@param** idDocName Identificador del documento en SGDA (docName)

\* **@return** Fichero físico del documento

\* **@throws** WdehAccesoSGDAException En caso de error con SGDA

\*/

**public** **static** **final** Fichero obtenerFicheroDocumento(String idDocName) **throws** WdehAccesoSGDAException {

// Recuperar el puerto de SGDA

OperacionesDocumento port = *getPuertoWsSGDA*();

// Parametros de conexion a SGDA

Conexion conexion = *crearParametroConexion*();

// Identificador del documento (docname) en SGDA

Identificador idDoc = **new** Identificador();

idDoc.setId(idDocName);

**try** {

// Llamada a SGDA para recuperar el fichero del documento

Fichero fichero = port.obtenerFicheroDocumento(conexion, idDoc);

**return** fichero;

} **catch** (ConexionException\_Exception e) {

**throw** **new** WdehAccesoSGDAException(e.getFaultInfo().getCodigo(), e.getFaultInfo().getMensaje());

} **catch** (ParametrosException\_Exception e) {

**throw** **new** WdehAccesoSGDAException(e.getFaultInfo().getCodigo(), e.getFaultInfo().getMensaje());

} **catch** (SGDAException e) {

**throw** **new** WdehAccesoSGDAException(e.getFaultInfo().getCodigo(), e.getFaultInfo().getMensaje());

}

}

/\*\*

\* Crea los parámetros de conexion de SGDA

\* **@return** Parámetros de conexion

\* **@throws** WdehAccesoSGDAException

\*/

**private** **static** **final** Conexion crearParametroConexion() **throws** WdehAccesoSGDAException {

// Parametros de conexion a SGDA

Conexion conexion = **new** Conexion();

// Recuperar el usuario virtual de WDEH de la JNDI correspondiente de acceso a SGDA

String virtualNote = WdehConfigJndi.*getVariable*(WdehConstantes.***WDEH\_REF\_VIRTUAL\_SGDA***);

// Establecer los parámetros de conexion

conexion.setUsuarioVirtual(virtualNote);

conexion.setRochade(WdehConstantes.***WDEH\_ROCHADE***);

conexion.setSilconUsuario(WdehConstantes.***WDEH\_SILCON***);

// Parámetros de conexion

**return** conexion;

}

/\*\*

\* Recupera el puerto del servicio web de SGDA

\* **@return** Puerto del servicio

\* **@throws** WdehAccesoSGDAException

\*/

**private** **static** **final** OperacionesDocumento getPuertoWsSGDA() **throws** WdehAccesoSGDAException {

// endpoint del servicio (estará en una JNDI)

String endpointWS = WdehConfigJndi.*getVariableURL*(WdehConstantes.***WDEH\_REF\_ENDPOINT\_SGDA***) + WdehConstantes.***SGDA\_WS\_DOCUMENTO***;

URL url = **null**;

**try** {

// URL del servicio web de SGDA

url = **new** URL(endpointWS + "?wsdl");

} **catch** (MalformedURLException e) {

**throw** **new** WdehAccesoSGDAException(WdehConstantes.***SGDA\_CODIGO\_ERROR\_URL\_WSDL*** ,WdehConstantes.***SGDA\_DESC\_ERROR\_URL\_WSDL*** + e.getMessage());

}

// Qname del servicio OperacionesDocumentoService

QName qname = **new** QName(WdehConstantes.***SGDA\_WS\_NAMESPACE***, WdehConstantes.***SGDA\_WS\_DOCUMENTO***);

OperacionesDocumentoService servicio = **new** OperacionesDocumentoService(url, qname);

OperacionesDocumento port = servicio.getOperacionesDocumentoPort();

// Devolver el puerto del servicio

**return** port;

}