CXF Web Service

Configuración

1. Descargar la última versión de Apache CXF en https://cxf.apache.org/download.html

2. Descomprimir los fuentes en un directorio conocido.

3. Acceder a Eclipse y configurar el acceso al directorio correspondiente en Windows / Preferences / Web Services / CXF 2.x Preferences,

seleccionando el directorio dentro de la solapa "CXF Runtime".

4. Acceder al directorio donde está instalado Java JDK en la ubicación %JAVA\_JDK%\jre\lib.

5. Buscar el archivo "jaxp.properties", de no existir crearlo.

6. Editar el archivo agregando la siguiente sentencia:

javax.xml.accessExternalSchema = all

7. Acceder al directorio donde está instalado Java JRE en la ubicación %JAVA\_JRE%\lib.

8. Repetir los pasos 5 y 6.

9. Reiniciar la máquina.

10. Listo.

Consideraciones al crear el servicio:

1. Seleccionar la clase que construirá el servicio y hacer botón derecho seleccionando New / Web Services / Web Service

2. Asegurarnos que Web service runtime diga "Apache CXF 2.x", continuar...

3. Puede o no crearse una interfaz asociada a la clase, continuar...

4. Marcar las anotaciones @WebMethod y @WebParam, continuar...

5. De las opciones de Java2WS seleccionar "Generate Server" dejando los que ya están seleccionados por defecto.

6. Desmarcar "Generate separate XSD for the types" para que toda la configuración quede en el archivo WSDL que se genere, finalizar.

7. Editar el archivo WSDL creado, cambiar el puerto al que se hace binding.

8. Editar la clase Server que se creo para utilizando la misma URL editada en el paso anterior.

9. Ejecutar la aplicación Java para dejar el servicio en funcionamiento.

10. Listo.

Consideraciones al crear el servicio cliente (utilizar un proyecto separado):

1. Copiar la URL a la que se hace binding agregando la query ?wsdl al final, utilizar un browser para verificar que

tenemos acceso a la WSDL.

2. Hacer botón derecho sobre el proyecto y seleccionar New / Web Service / Web Service Client

3. Utilizar la URL del paso 1.

4. Verificar las clases generadas y determinando cómo consumir el servicio de acuerdo a su definición.

5. Listo.

Como usar:

En el cliente se va a poder utilizar la clase NumerosService, la cual me hace de factory para el Numeros, y de ahi uso las funciones de Numeros como si tuviera el objeto.

NumerosService ss = new NumerosService(wsdlURL, SERVICE\_NAME);

Numeros port = ss.getNumerosPort();

{

System.out.println("Invoking getNumero...");

int \_getNumero\_\_return = port.getNumero();

System.out.println("getNumero.result=" + \_getNumero\_\_return);

}

Axis Web Service

Configuración

1. Descargar la versión 1.6.2 de Apache Axis2 en https://archive.apache.org/dist/axis/axis2/java/core/1.6.2/

2. Descomprimir los fuentes en un directorio conocido.

3. Acceder a Eclipse y configurar el acceso al directorio correspondiente en Windows / Preferences / Web Services / Axis2 Preferences,

seleccionando el directorio dentro de la solapa "Axis2 Runtime".

4. Listo.

Consideraciones al crear los servicios:

Es importante que el parámetro "Dynamic Web Module" sea 2.5, tanto en el proyecto que crea el servicio como en aquel que lo

consume.

Creo que es igual al de CXF

FactorialWSService service = new FactorialWSService();

IFactorialWS factorial = service.getFactorialWSPort();

factorial.getFactorial(Integer.valueOf(request.getParameter("nro")))

JAX Web Services

Ejecutar las siguientes líneas de comando en una ventana del sistema estando ubicados en el directorio del proyecto, seguir

estos pasos siempre que no se haya creado el servicio cliente.

cd %PROJECT\_HOME%/src

wsimport -s . http://localhost:{PORT}/FactorialJAXWS/FactorialWS?wsdl

NOTA 1: Reemplazar {PORT} de acuerdo al puerto en el cual se publicó el servicio. En nuestro caso sería 8091.

NOTA 2: Esto generará el servicio para ser consumido, no se requiere ejecutar más de una vez, a menos que haya un cambio en

el servicio.

REST Web Services

Configuración

1. Descargar la última versión de Jersey en https://jersey.java.net/download.html

2. Descomprimir los fuentes en un directorio conocido.

3. Descargar el archivo JAR de Jackson JAX-RS Provider en https://mvnrepository.com/artifact/com.fasterxml.jackson.jaxrs/jackson-jaxrs-json-provider/2.8.3

junto con los archivos JAR dependientes

4. Copiar el archivo dentro del directorio ext de Jersey.

5. Luego en Windows / Preferences / Validation deshabilitar las opciones de "Classpath Dependency Validation".

5. Listo.

Consideraciones al crear el servicio:

1. Al crear el proyecto, hacer clic en el botón "Modify" de la configuración del mismo, seleccionar el project facet "JAX-RS (REST Web Services)"

2. En el paso "Web Module" seleccionar la creación del archivo web.xml.

3. En el paso "JAX-RS Capabilities", seleccionar en la lista "Type" la opción "User Library". Si la librería aun no fue creada, hacer clic en el primer ícono a la

derecha.

Hacer clic en el botón "New...", poner de nombre "JAX-RS <Version> Library" a la librería y marcar que sea una clase del sistema. Luego seleccionar todos los archivos JAR del directorio descargado en la configuración.

Para ello, utilizar el botón "Add External JARs...". Aceptar los cambios.

4. Marcar la librería.

5. Hacer clic en "Update Deployment Descriptor", indicar en el campo "JAX-RS servlet name" el valor JAXRSServlet y en el campo "JAX-RS servlet class name" el valor

"org.glassfish.jersey.servlet.ServletContainer".

A su vez, eliminar en "URL mapping patterns" el valor actual y agregar el valor "/rest/\*"

6. Finalizar para que se cree el proyecto.

7. Editar el archivo web.xml agregar a la definición del Servlet JAX-RS, el parámetro:

<init-param>

<param-name>jersey.config.server.provider.packages</param-name>

<param-value>ar.edu.ubp.das.resources,com.fasterxml.jackson.jaxrs</param-value>

</init-param>

NOTA: El valor "ar.edu.ubp.das.resources" es el paquete que contiene las clases desarrolladas por lo que puede cambiar su nombre.

8. Listo.