Desarrollo de Servicios Web 1

Tema Nº9:Servicios Web con Apache CXF

Indicador de logro Nº9:Desarrolla servicios utilizando JAX-WS a través del framework Apache CXF.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº9:**

SERVICIOS WEB CON APACHE CXF

**Subtema 1:**

FRAMEWORK CXF

**MARCO TEÓRICO**

Es un framework de servicios de software libre. CXF nos ayuda a construir y desarrollar servicios utilizando JAX-WS como API de programación.

Estos servicios pueden “hablar” una gran variedad de protocolos como SOAP, XML/HTTP, HTTP RESTful o CORBA, y pueden trabajar sobre transportes como HTTP, JMS o JBI. Sus características principales son:

* **Soporte para estándares de Servicios Web**

CXF soporta varios estándares de servicios web incluyendo a SOAP, el Perfil Básico, WSDL, WS-Addressing, WS-Policy, WS-ReliableMessaging y WS-Security.

* **Interfaces**

CXF da soporte a varios modelos de programación como “interfaz”. CXF implementa el API JAX-WS. También incluye una interfaz simple que permite crear clientes y endpoints sin utilizar anotaciones. CXF soporta el desarrollo por contrato primero con WSDL, y el desarrollo por código primero comenzando desde Java.

* **Facilidad de uso**

CXF está diseñado para ser intuitivo y fácil de usar. Hay APIs simples para construir servicios comenzando por el código, plug-ins de Maven para integrar esta herramienta, soporte para el API JAX-WS, soporte de XML de Spring 2.0 y posteriores, para facilitar la configuración.

* **Soporte para protocolos binarios y legacy**

CXF fue diseñado para proveer una arquitectura extensible que no solo soporte XML sino también otros binding no XML, como JSON y CORBA, en combinación con cualquier tipo de transporte.

**Librería Apache CXF**

La librería Apache CXF, la última versión se descarga de la página web http://cxf.apache.org/, una vez descargada, se descomprime los ficheros en una ruta conocida, esta ruta se usará para configurar el plugin de eclipse CXF.



**1. MATERIALES**

Para la experiencia a realizar se requiere lo siguiente:

1. EQUIPO

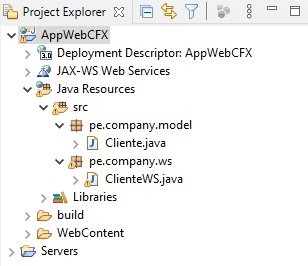
Se requiere tener un equipo de cómputo (PC o laptop).

1. HERRAMIENTAS

* Java Developer Kit (JDK) version 8.x
* Eclipse IDE
* Apache TomCat 7.x.x
* apache-cxf-3.1.4
* SoapUI 5.5

**2. PROCEDIMIENTO**

Se tiene el proyecto Web:



Clase VO:

package pe.company.model;

public class Cliente

{

private int id\_cliente;

private String empresa;

private String vendedor;

private double credito;

public Cliente() {}

public Cliente(int id\_cliente, String empresa, String vendedor, double credito) {

this.id\_cliente = id\_cliente;

this.empresa = empresa;

this.vendedor = vendedor;

this.credito = credito;

}

//Getters and Setters

}

Clase WS:

Está clase vendría a ser la implementación

package pe.company.ws;

import java.util.ArrayList;

import java.util.Collection;

import pe.company.model.Cliente;

public class ClienteWS

{

private static final Collection<Cliente> *itemsCliente*=new ArrayList();

static

{

Cliente cliente1=new Cliente(1021,"Acme Mfg.","Bill Adams",50000.0);

Cliente cliente2=new Cliente(1022,"Zetacorp","Larry Fitch",50000.0);

Cliente cliente3=new Cliente(1023,"Ace International","Tom Snyder",35000.0);

Cliente cliente4=new Cliente(1024,"Peter Brothers","Nancy Angelli",40000.0);

*itemsCliente*.add(cliente1);

*itemsCliente*.add(cliente2);

*itemsCliente*.add(cliente3);

*itemsCliente*.add(cliente4);

}

public ClienteWS() {

}

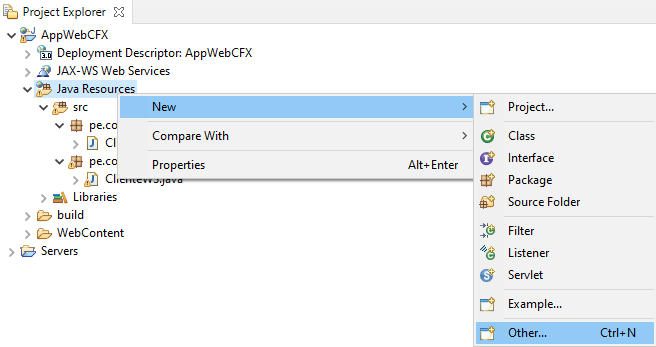
public Collection<Cliente> findAll(int codigo) {

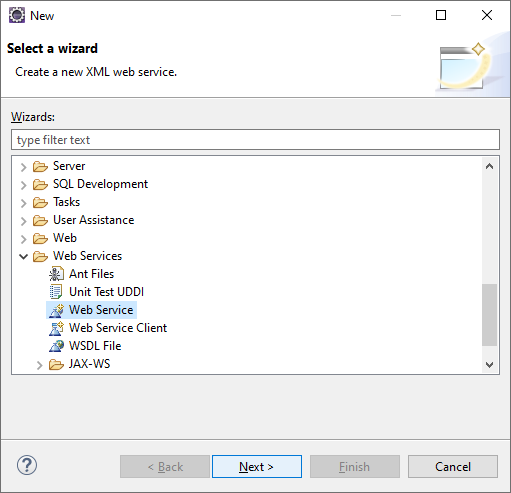
return *itemsCliente*;

}

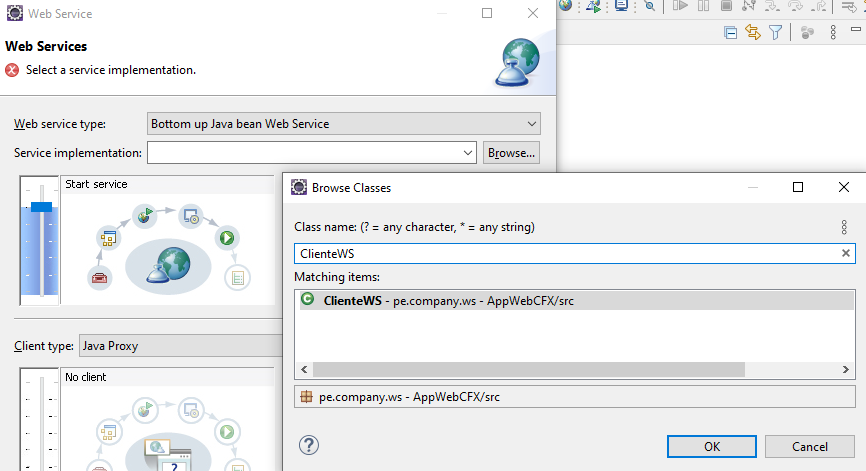
}

Ahora se pasa a crear la interface, esta interface vendría a ser el punto de ingreso al Web Service:



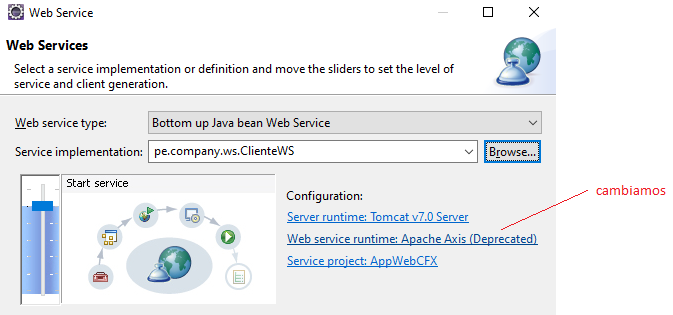


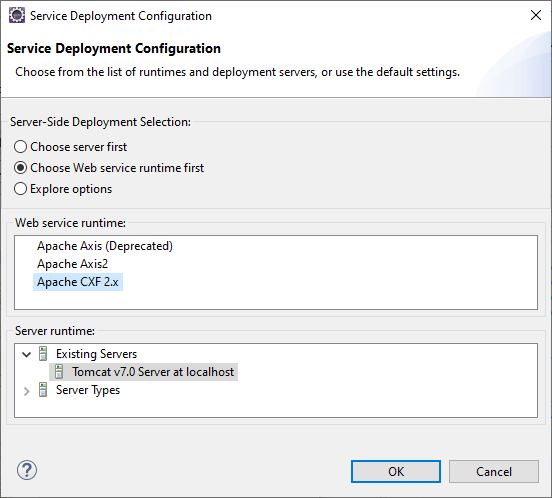
Se va a indicar cuál será la implementación, presionamos en Browser y seleccionamos:

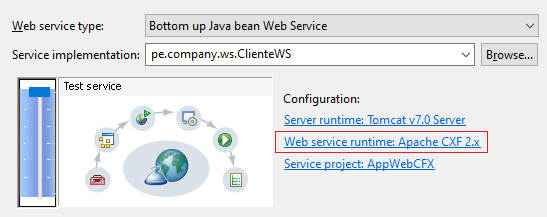


OK

Ahora escogeremos Apache CXF en:





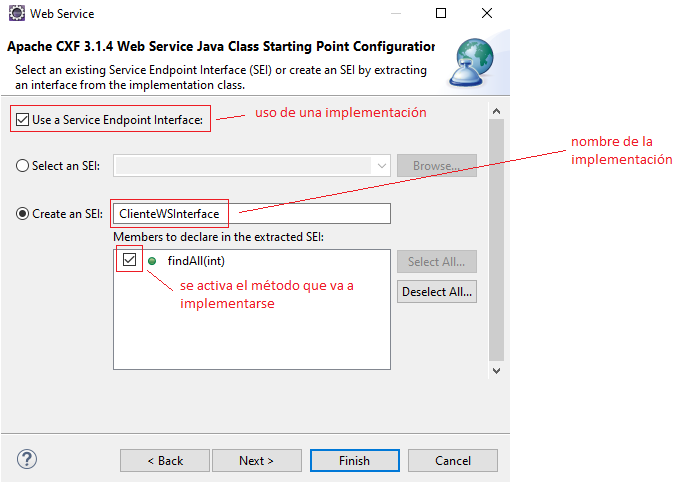


Next 🡪

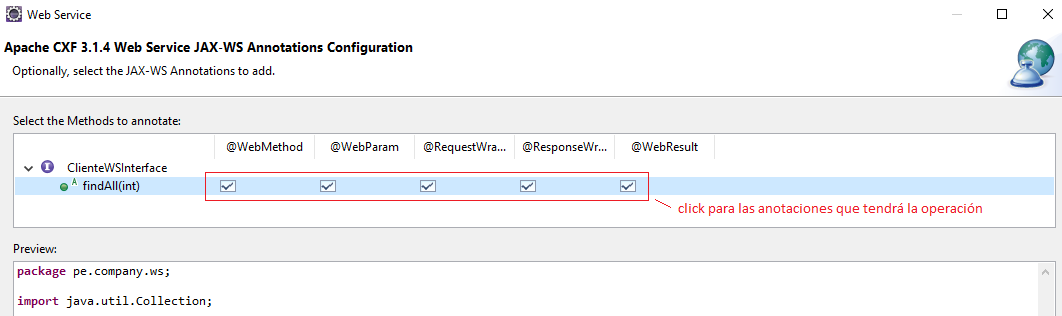
Está instalando CXF al proyecto Web:

Nota: En caso le genere una excepción es debido que no agregó el entorno de tiempo de ejecución CXF, debe pasar a configurar Apache CXF con su instructor.

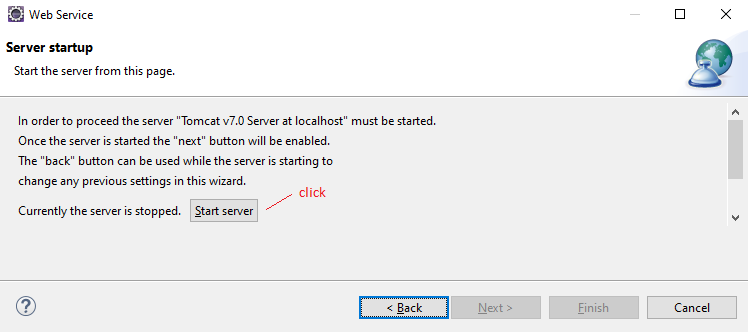
Luego se tiene que indicar que se va a usar una implementación:



Next (Aparecerá la ventana de anotaciones) 🡪

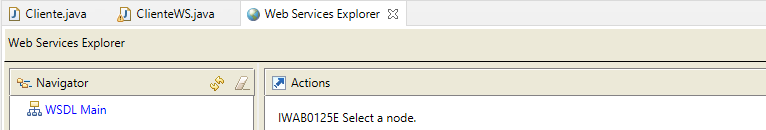


Next 🡪 Next (Aparecerá la ventana que nos indica que debemos tener iniciado el Tomcat) 🡪

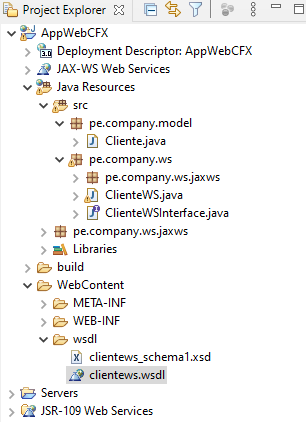


Next 🡪 Finish (Nos mostrará el Web Service Explorer) 🡪

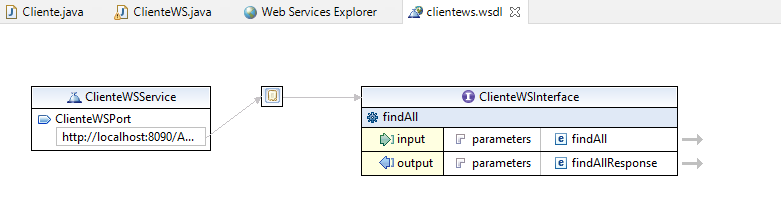
Nos muestra el acceso al Web Service



Se tiene en el proyecto el WSDL:



Se observa el WSDL en vista diseño:



Se observa la creación de la interface:

package pe.company.ws;

import java.util.Collection;

import javax.jws.\*;

import javax.xml.ws.\*;

import pe.company.model.Cliente;

@WebService(name = "ClienteWSInterface", targetNamespace = "http://ws.company.pe/")

public interface ClienteWSInterface {

@WebMethod(operationName = "findAll", action = "urn:FindAll")

@RequestWrapper(className = "pe.company.ws.jaxws.FindAll",

localName = "findAll", targetNamespace = "http://ws.company.pe/")

@ResponseWrapper(className = "pe.company.ws.jaxws.FindAllResponse",

localName = "findAllResponse", targetNamespace = "http://ws.company.pe/")

@WebResult(name = "return")

Collection<Cliente> findAll(@WebParam(name = "arg0") int codigo);

}

Y también se genera la clase ClienteWS con su implementación de la interface ClienteWSInterface:

package pe.company.ws;

import java.util.ArrayList;

import java.util.Collection;

import javax.jws.WebService;

import pe.company.model.Cliente;

@WebService(targetNamespace = "http://ws.company.pe/",

endpointInterface = "pe.company.ws.ClienteWSInterface",

portName = "ClienteWSPort",

serviceName = "ClienteWSService")

public class ClienteWS implements ClienteWSInterface

{

private static final Collection<Cliente> *itemsCliente*=new ArrayList();

static

{

Cliente cliente1=new Cliente(1021,"Acme Mfg.","Bill Adams",50000.0);

Cliente cliente2=new Cliente(1022,"Zetacorp","Larry Fitch",50000.0);

Cliente cliente3=new Cliente(1023,"Ace International","Tom Snyder",35000.0);

Cliente cliente4=new Cliente(1024,"Peter Brothers","Nancy Angelli",40000.0);

*itemsCliente*.add(cliente1);

*itemsCliente*.add(cliente2);

*itemsCliente*.add(cliente3);

*itemsCliente*.add(cliente4);

}

public ClienteWS() {

}

public Collection<Cliente> findAll(int codigo) {

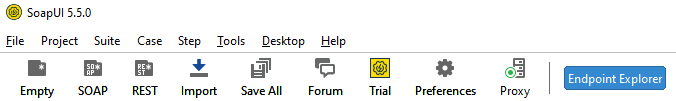
return *itemsCliente*;

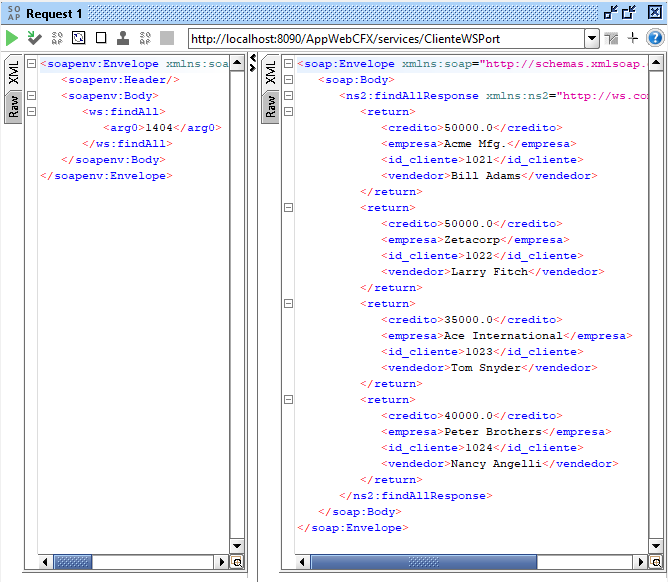
}

}

Para verificar el Web Service se pasa hacer la prueba unitaria con el SoapUI:







**ACTIVIDAD VIRTUAL:**

1. **CUESTIONARIO TÉCNICO**

Revisa y analiza el tema desarrollado en la presente sesión, luego responde las siguientes preguntas propuestas:

* ¿Usar CXF qué beneficios trae?
* ¿Es CXF una implementación que ofrece alto rendimiento en comparación con Axis?
* ¿Mencione y describa algunas anotaciones utilizadas por CXF?
* ¿CXF puede integrarse con Spring, dar una breve explicación?
* Mediante Eclipse IDE, Base de datos, Apache Tomcat y Apache CXF crear web services para las operaciones a la BD. Realizar pruebas unitarias.

1. **ENLACES DE REFERENCIA**

* http://www.javahispano.org/antiguo\_javahispano\_org/2009/1/25/servicios-web-con-apache-cxf.html
* http://dpinya.blogspot.com/2013/11/apache-cxf-invocacion-de-servicios-web.html

1. **CONCLUSIONES DE LA EXPERIENCIA**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_