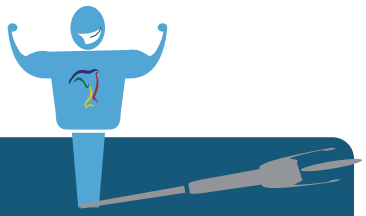
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

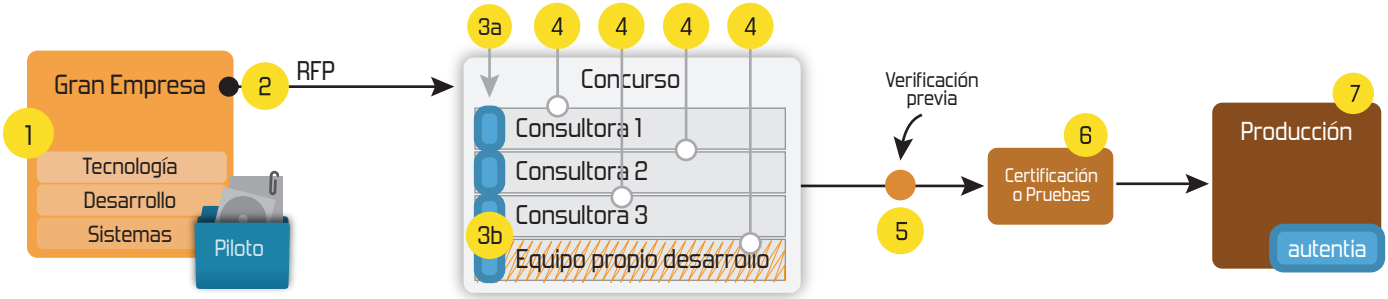
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

- Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery
- Gestor portales (Liferay)
- Gestor de contenidos (Alfresco)
- Aplicaciones híbridas
- Control de autenticación y acceso (Spring Security)
- UDDI
- Web Services
- Rest Services
- Social SSO
- SSO (Cas)
- JPA-Hibernate, MyBatis
- Motor de búsqueda empresarial (Solr)
- ETL (Talend)
- Dirección de Proyectos Informáticos.
- Metodologías ágiles
- Patrones de diseño
- TDD
- Tareas programadas (Quartz)
- Gestor documental (Alfresco)
- Inversión de control (Spring)
- BPM (jBPM o Bonita)
- Generación de informes (JasperReport)
- ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Carlos León Villamayor

Consultor tecnológico de desarrollo de proyectos informáticos. Constructor de Adictos Al Trabajo 2.0

Ingeniero Técnico en Informática (cursando grado superior)

Ver todos los tutoriales del autor

PDF Creator
Last Updated: Aug 23, 2011
License: Free
OS: Windows 7/Vista/2003/XP/2000/NT
Requirements: No special requirements
Download

Fecha de publicación del tutorial: 2012-04-24

Tutorial visitado 1 veces Descargar en PDF

Primeros pasos con Apache CXF

0. Índice de contenidos.

- 1. Entorno
- 2. Introducción
- 3. Configuración de eclipse
- 4. Aproximaciones. Code First
- 5. Aproximaciones. Contract First
- 6. Conclusiones

1. Entorno

Este tutorial está desarrollado usando el siguiente entorno:

- Hardware: Portátil MacBook Pro 2Ghz Intel Core i7 (4 núcleos) 8Gb de RAM
- Sistema Operativo: Mac OS X 10.7.2 (Lion)
- Eclipse Indigo (Revisar tutorial de Alex para su instalación)
- Apache Tomcat 7 (instalado e integrado en eclipse)
- Versión de java SDK 6 instalada en el sistema

2. Introducción

En este tutorial veremos como realizar servicios con Apache CXF basados en JAX-WS (http://en.wikipedia.org/wiki/JAX-WS) utilizando las WST de eclipse.

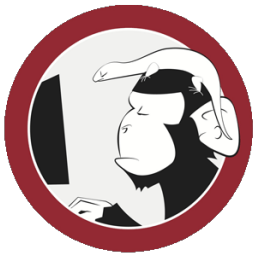
Apache CXF es un framework de Web Services que nace de la combinación de dos proyectos open source Celtix y XFire. Sus principales caracteristicas son (sacado de la wikipedia):

- Web Services Standards
  - SOAP
  - WS-Addressing
  - WS-Policy
  - WS-ReliableMessaging
  - WS-SecureConversation
  - WS-Security
  - WS-SecurityPolicy
- JAX-WS API for Web service development
  - Java first support
  - WSDL first tooling
- JAX-RS (JSR 311 1.1) API for RESTful Web service development
- JavaScript programming model for service and client development
- Maven tooling
- CORBA support
- HTTP and JMS transport layers
- Embeddable Deployment:
  - ServiceMix or other JBI containers
  - Geronimo or other Java EE containers
  - Tomcat or other servlet containers
  - OSGi
- Reference OSGi Remote Services implementation

3. Configuración de eclipse

Primero de todo debemos configurar eclipse con la versión de CXF que vayamos a utilizar:

Catálogo de servicios Autentia



Síguenos a través de:

Últimas Noticias

» Comentando el Libro: Lean StartUp (el método)

» VIII Autentia Cycling Day

» ¿Por qué una empresa puede dar unos grandes beneficios y tener que despedir gente?

» Técnico en preventa: entre técnico y comercial

» Rotación en la empresa y el teorema de la niña mona

Histórico de noticias

Últimos Tutoriales

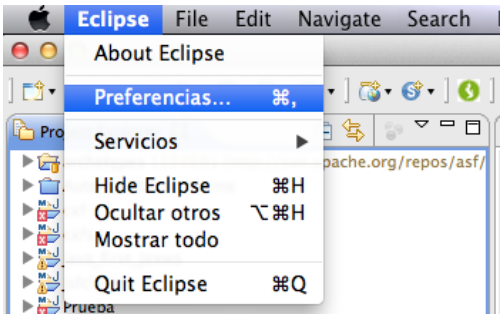
» Cómo usar el GPS en nuestras aplicaciones iOS 5 (iPhone, iPad, ...)

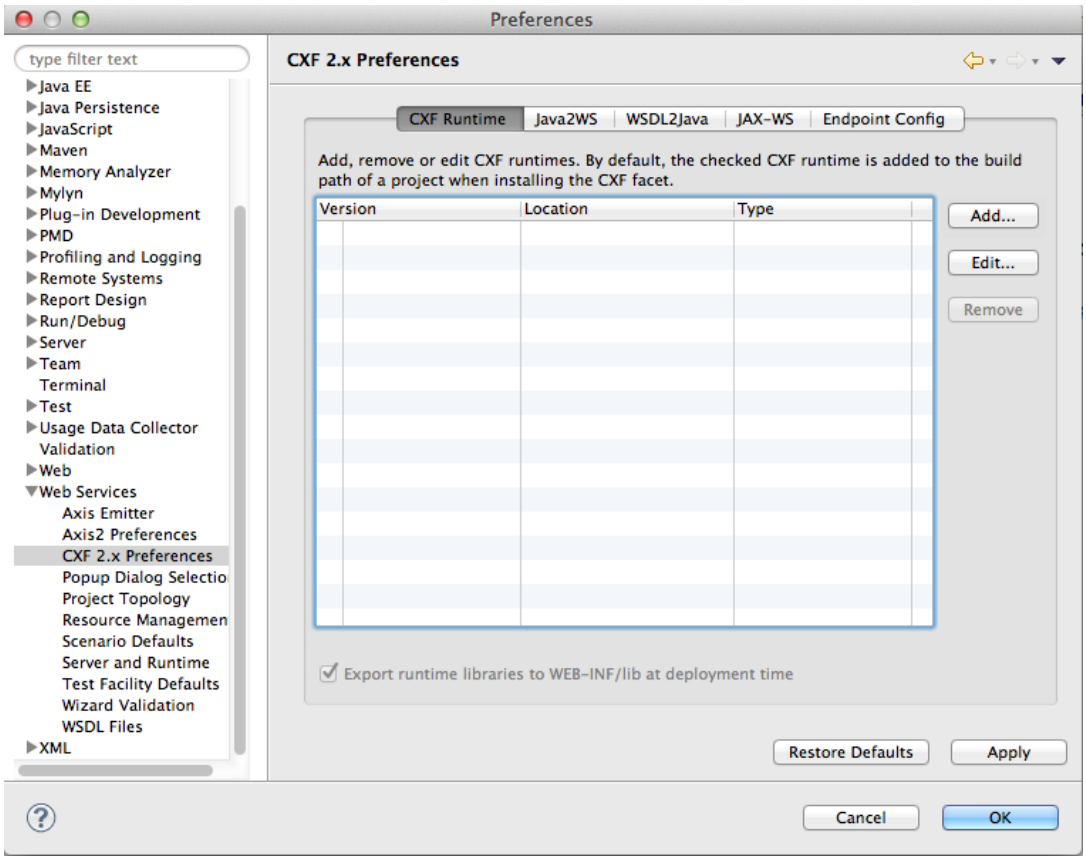
» Modelando a Pólux en Blender (1/3)

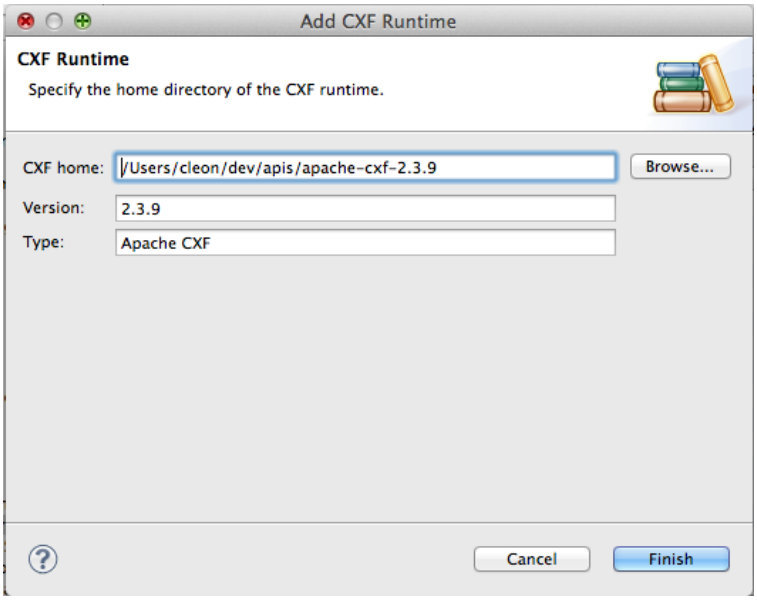
» Balanceando la carga de MySQL con HAProxy

» Como internacionalizar nuestras aplicaciones iOS 5 (iPhone, iPad, ...)

» Subir una aplicación a BlackBerry App World







Últimos Tutoriales del Autor

» Primeros pasos con ServiceMix 4.4.0

» Cluster en Glassfishv3 usando mod_jk para el balanceo de peticiones

» Tomcat Cluster con JSF

» Primeros pasos con Spring Web Flow 2

» Monitorizando estado de servidores JEE con Nagios y JMX4Perl

Últimas ofertas de empleo

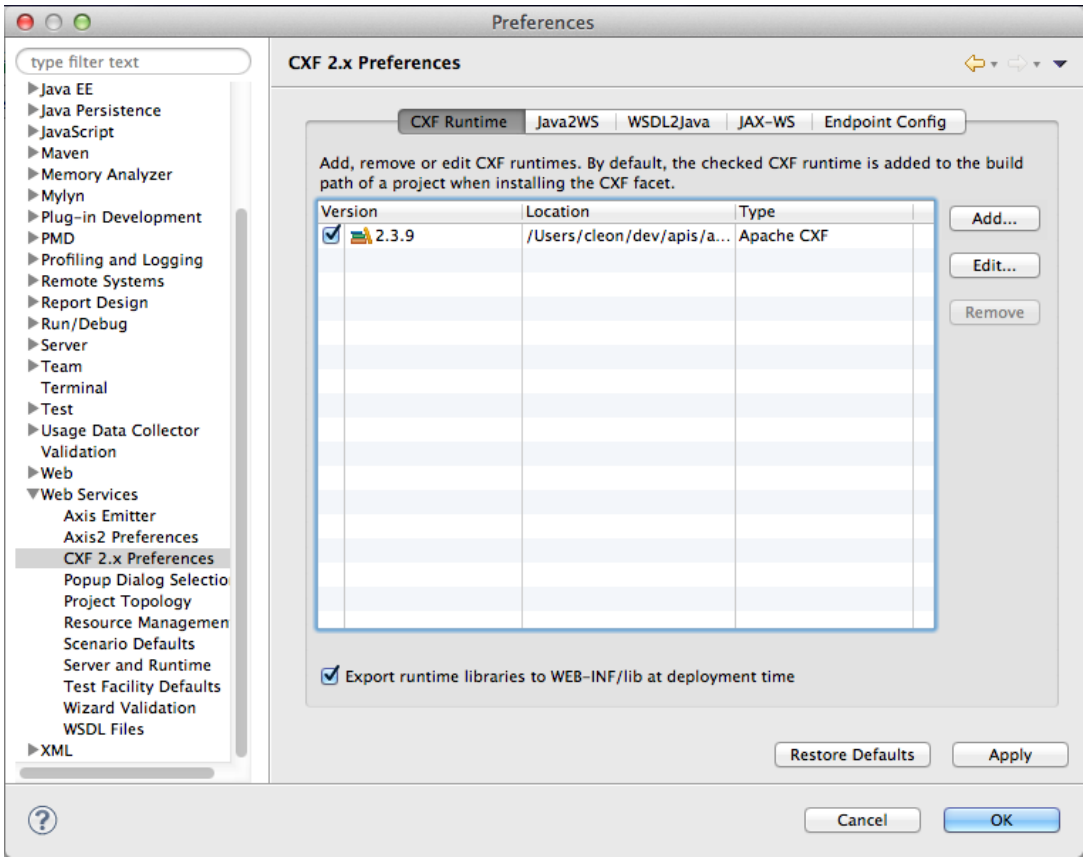
2011-09-08
Comercial - Ventas - MADRID.

2011-09-03
Comercial - Ventas - VALENCIA.

2011-08-19
Comercial - Compras - ALICANTE.

2011-07-12
Otras Sin catalogar - MADRID.

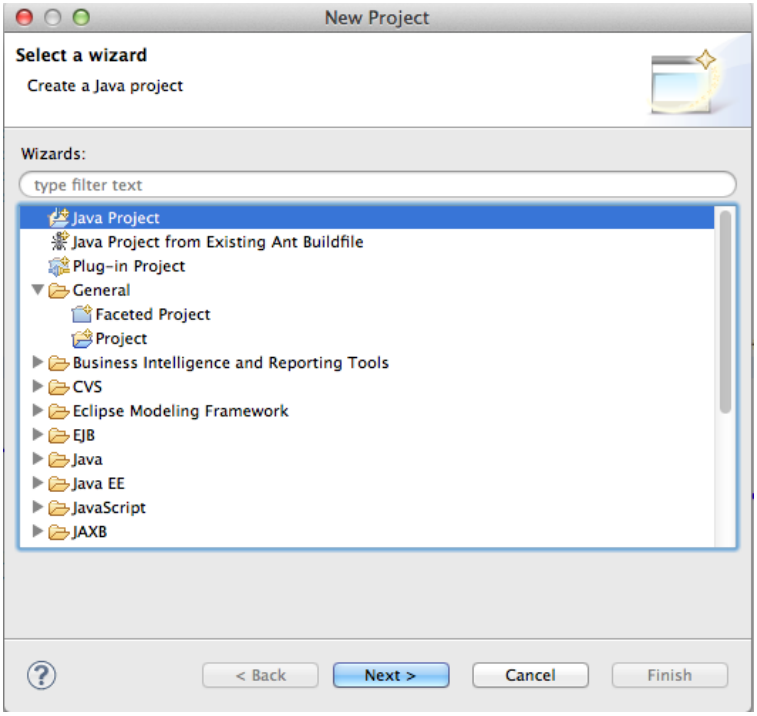
2011-07-06
Otras Sin catalogar - LUGO.

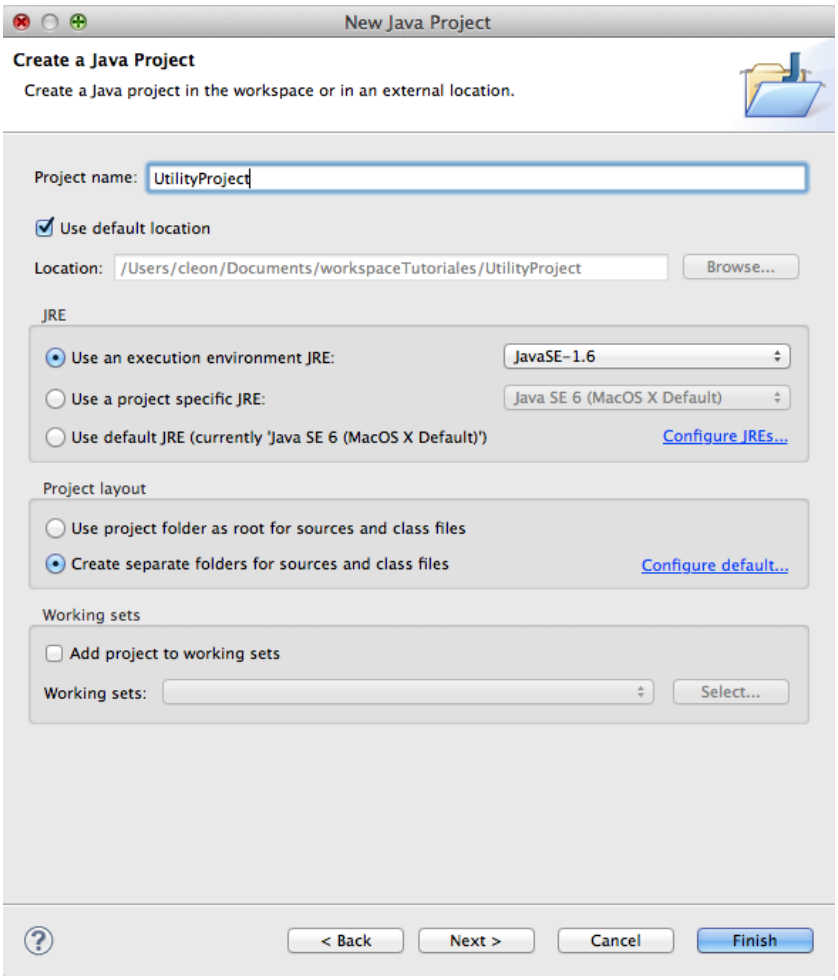


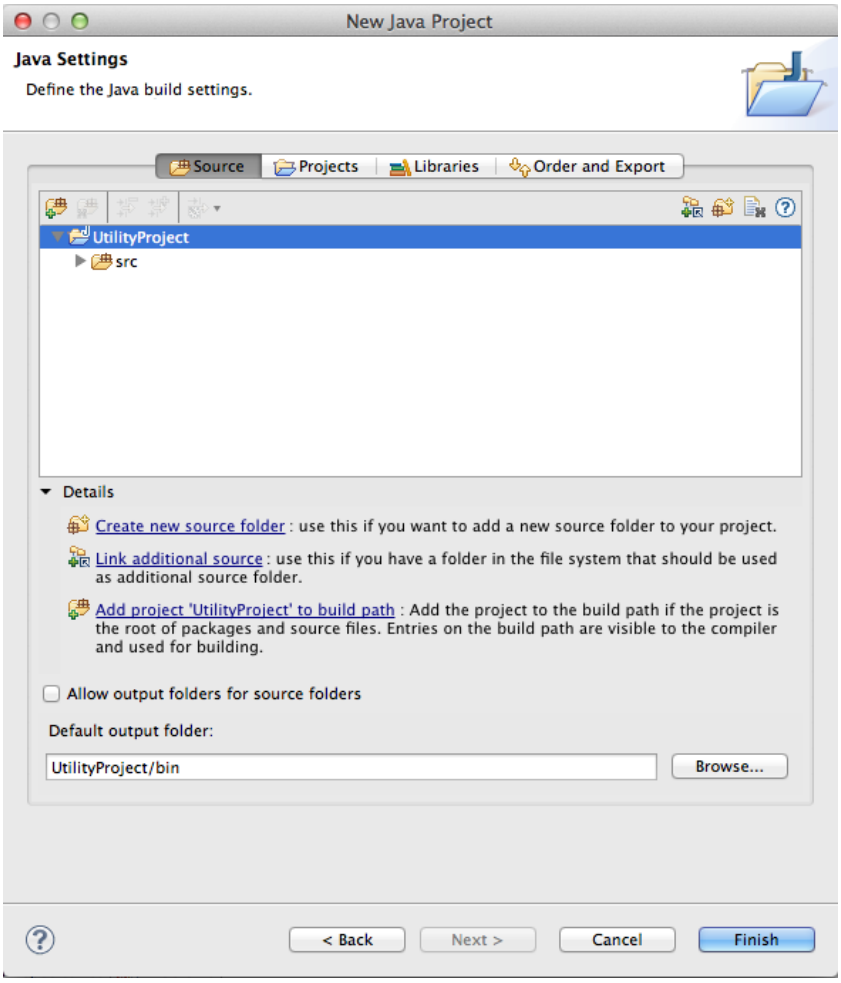
Una vez configurado podremos utilizarlo en el asistente de creación de webservices.

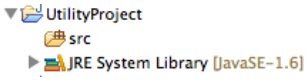
4. Aproximaciones. Code First

En esta primera aproximación partimos de una clase que expondrá los métodos que le especifiquemos explicitamente en el webservice. Para ello creamos un nuevo proyecto java y codificamos una interfaz (a la que llamaremos SEI) con su implementación, y una entidad Persona. Para la persistencia utlizaremos un HashMap que almacenará las personas:




New Java Project
Create a Java Project
Create a Java project in the workspace or in an external location.
Project name: UtilityProject
Use default location
Location: /Users/cleon/Documents/workspaceTutoriales/UtilityProject
Browse...
JRE
Use an execution environment JRE: JavaSE-1.6
Use a project specific JRE: Java SE 6 (MacOS X Default)
Use default JRE (currently 'Java SE 6 (MacOS X Default)')
Configure JREs...
Project layout
Use project folder as root for sources and class files
Create separate folders for sources and class files
Configure default...
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Cancel
Finish


New Java Project
Java Settings
Define the Java build settings.
Source
Projects
Libraries
Order and Export
UtilityProject
src
Details
Create new source folder : use this if you want to add a new source folder to your project.
Link additional source : use this if you have a folder in the file system that should be used as additional source folder.
Add project 'UtilityProject' to build path : Add the project to the build path if the project is the root of packages and source files. Entries on the build path are visible to the compiler and used for building.
Allow output folders for source folders
Default output folder:
UtilityProject/bin
Browse...
< Back
Next >
Cancel
Finish


UtilityProject
src
JRE System Library [JavaSE-1.6]

Entidad:

```java
package com.autentia.cxf;

import java.io.Serializable;

/**
 * @author cleon
 *
 */
public class Persona implements Serializable{
    private static final long serialVersionUID = 1677577045608086848L;
    private String nombre;
    private String apellidos;
    public Persona() {
    }
    /**
     * @return the nombre
     */
    public String getNombre() {
        return nombre;
    }
    /**
     * @param nombre the nombre to set
     */
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    /**
     * @return the apellidos
     */
    public String getApellidos() {
        return apellidos;
    }
    /**
     * @param apellidos the apellidos to set
     */
    public void setApellidos(String apellidos) {
        this.apellidos = apellidos;
    }

}
```

Singleton que mantiene la persistencia de las personas:

```java
package com.autentia.cxf;

import java.util.HashMap;
import java.util.Map;

/**
 * @author cleon
 *
 */
public class Singleton {
    private static final Singleton instance = new Singleton();
    private Map<String,Persona> cuentas = new HashMap<String, Persona>();
    private Singleton(){}
    public static Singleton getInstance() {
        return instance;
    }
    public Map<String,Persona> getPersonas(){
        return cuentas;
    }
}
```

Interfaz:

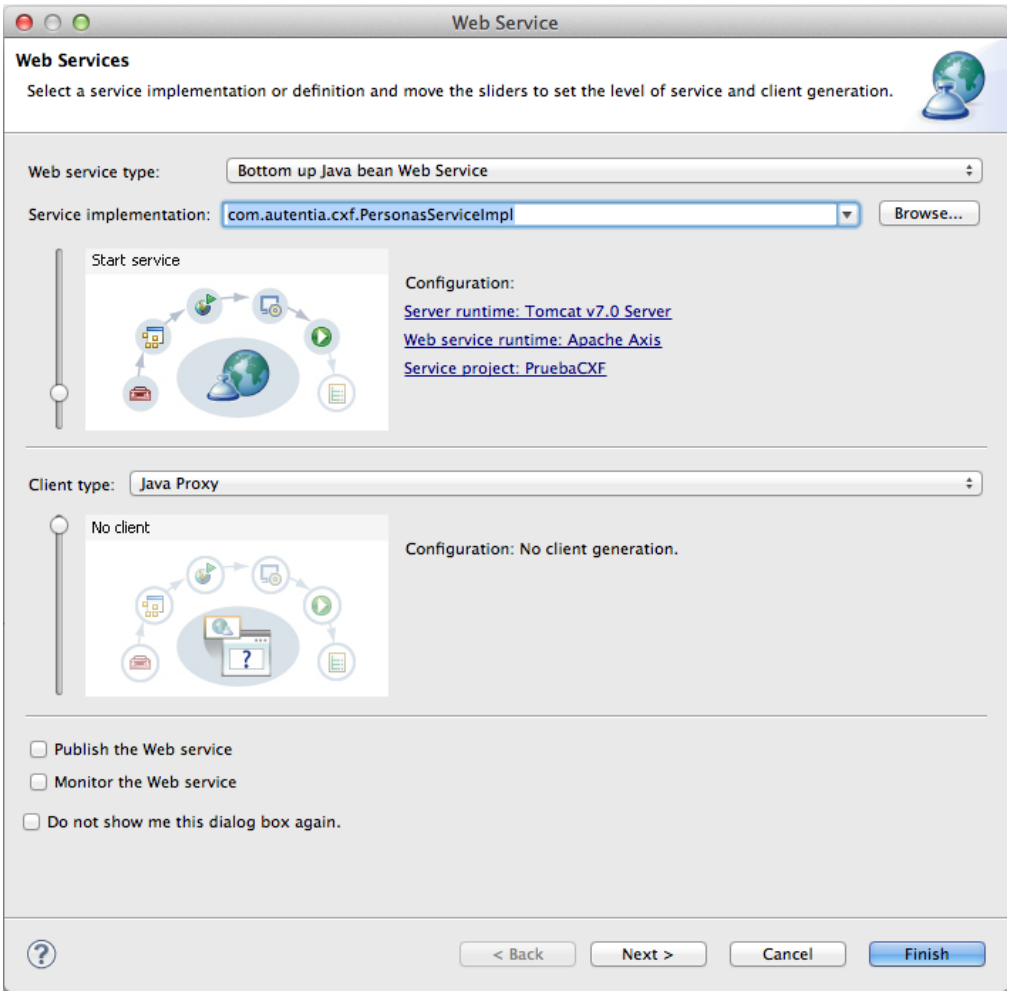
```java
package com.autentia.cxf;

import java.util.List;
/**
 * @author cleon
 *
 */
public interface PersonasSEI {
    /**
     * Añade una persona al listado
     * @param persona
     */
    void addPersona(Persona persona);
    /**
     * Devuelve una persona y en caso de no encontrarla lanza una excepcion
     * @param nombre
     * @return
     * @throws Exception
     */
    Persona getPersona(String nombre) throws Exception;
    /**
     * Obtiene el listado de personas almacenado
     * @return
     */
    List<Persona> getPersonas();
    /**
     * Elimina una persona del listado y si no existe lanza una excepcion
     * @param nombre
     * @throws Exception
     */
    void removePersona(String nombre) throws Exception;
}
```

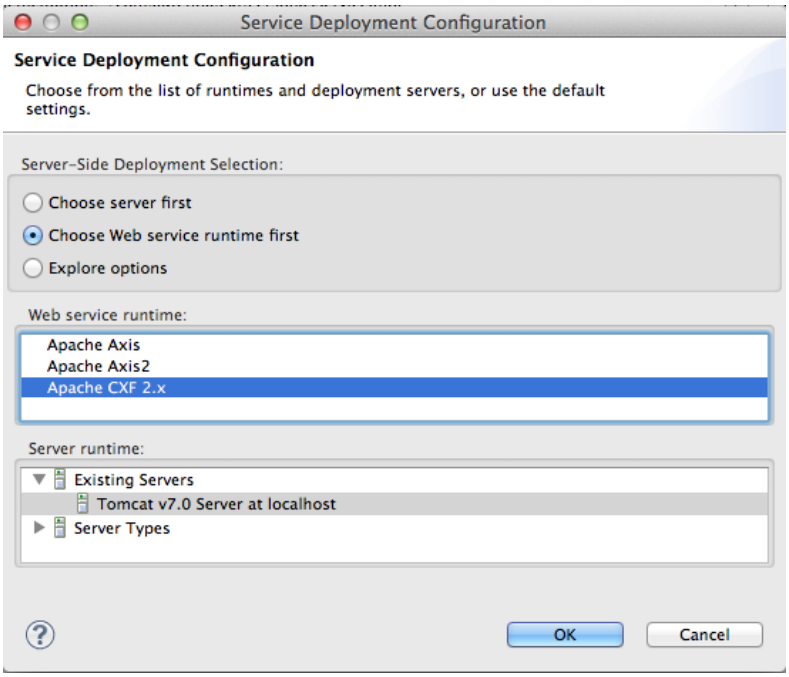
Implementación:

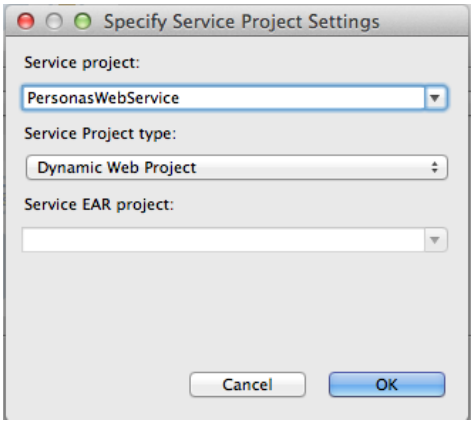
```java
package com.autentia.cxf;

import java.util.ArrayList;
```

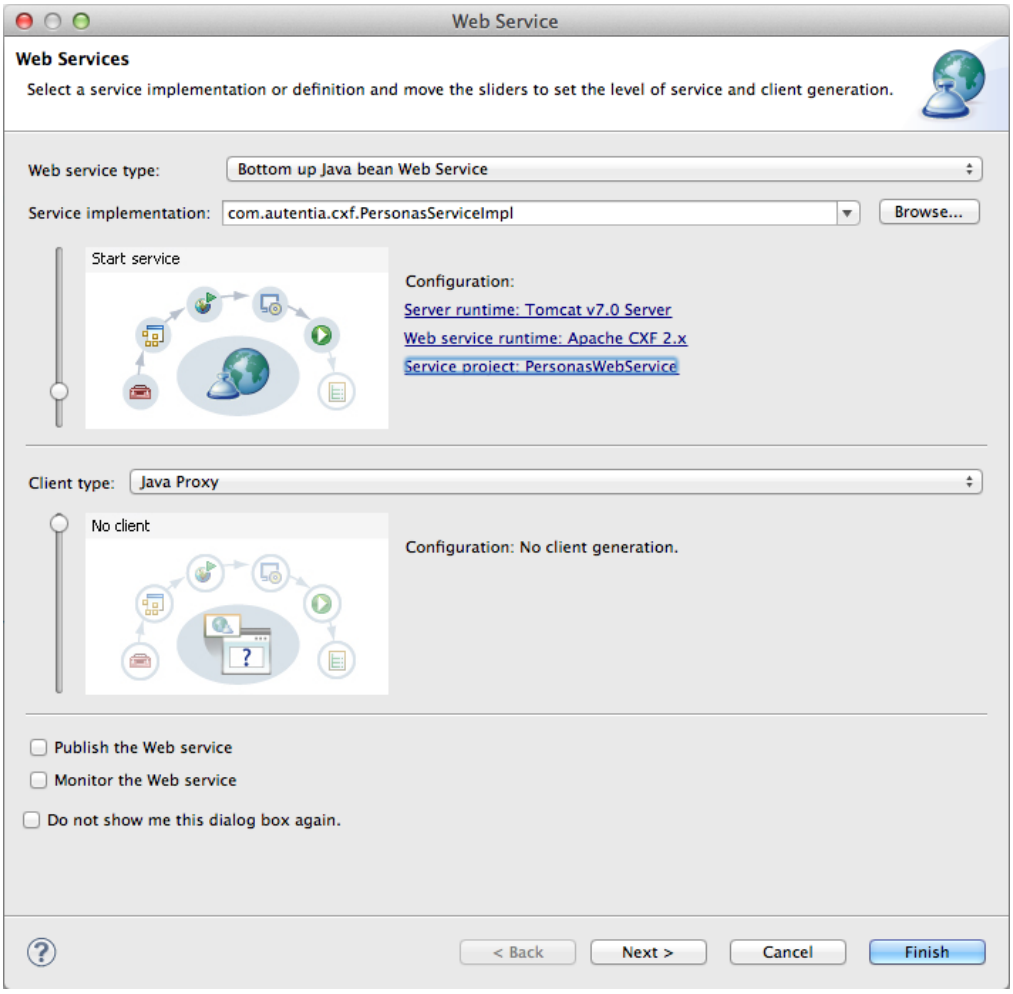
```java
import java.util.List;

/**
 * @author cleon
 *
 */
public class PersonasServiceImpl implements PersonasSEI {

    /* (non-Javadoc)
     * @see com.autentia.cxf.PersonaSEI#addPersona(com.autentia.cxf.Persona)
     */
    @Override
    public void addPersona(Persona persona) {
        Singleton.getInstance().getPersonas().put(persona.getNombre(), persona);
    }

    /* (non-Javadoc)
     * @see com.autentia.cxf.PersonaSEI#getPersona(java.lang.String)
     */
    @Override
    public Persona getPersona(String nombre) throws Exception {
        Persona persona = Singleton.getInstance().getPersonas().get(nombre);
        if (persona==null){
            throw new Exception("La persona buscada no existe");
        }
        return persona;
    }

    /* (non-Javadoc)
     * @see com.autentia.cxf.PersonaSEI#getPersonas()
     */
    @Override
    public List<Persona> getPersonas() {
        return new ArrayList<Persona>(Singleton.getInstance().getPersonas().values());
    }

    /* (non-Javadoc)
     * @see com.autentia.cxf.PersonaSEI#removePersona(java.lang.String)
     */
    @Override
    public void removePersona(String nombre) throws Exception {
        Persona persona = Singleton.getInstance().getPersonas().get(nombre);
        if (persona==null){
            throw new Exception("La persona buscada no existe");
        }else{
            Singleton.getInstance().getPersonas().remove(nombre);
        }

    }

}
```

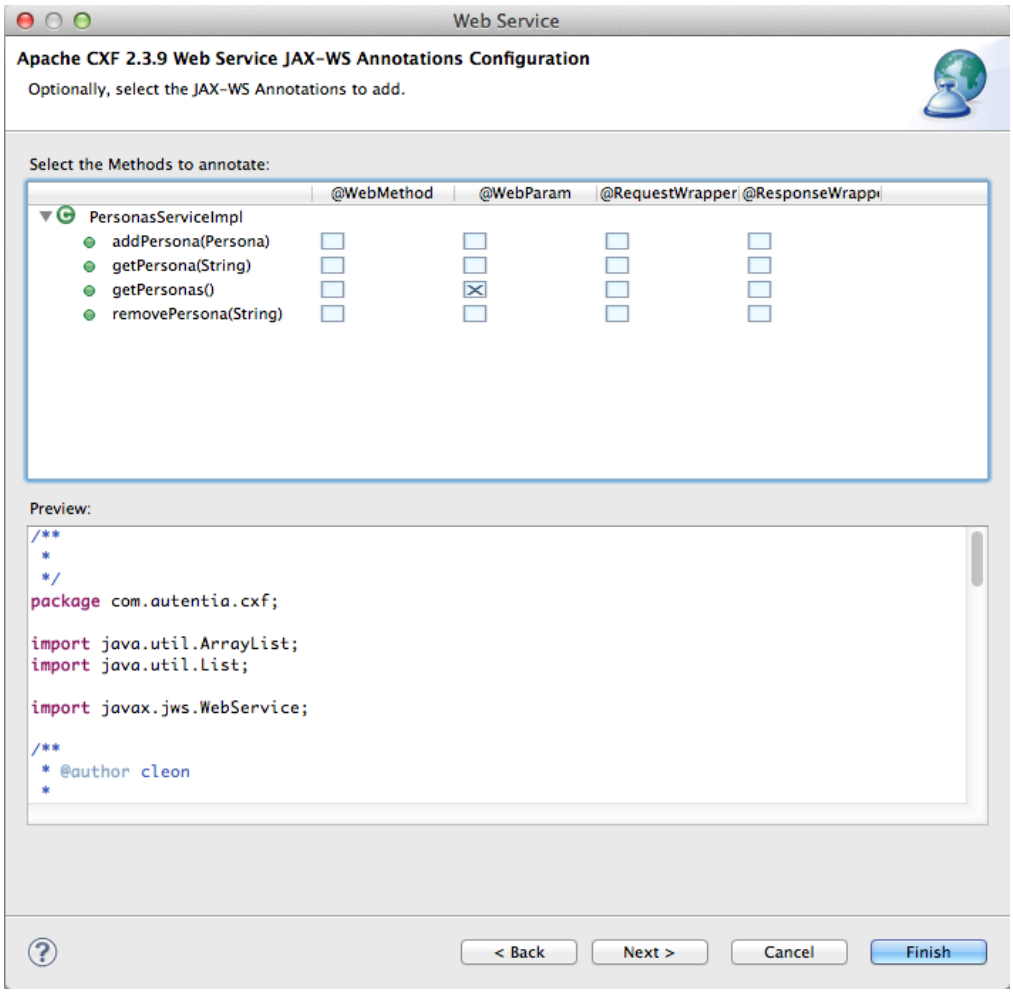
Una vez definidas las clases, hacemos click derecho sobre la implementación y webservices --> createwebservice y seguimos los siguientes pasos:

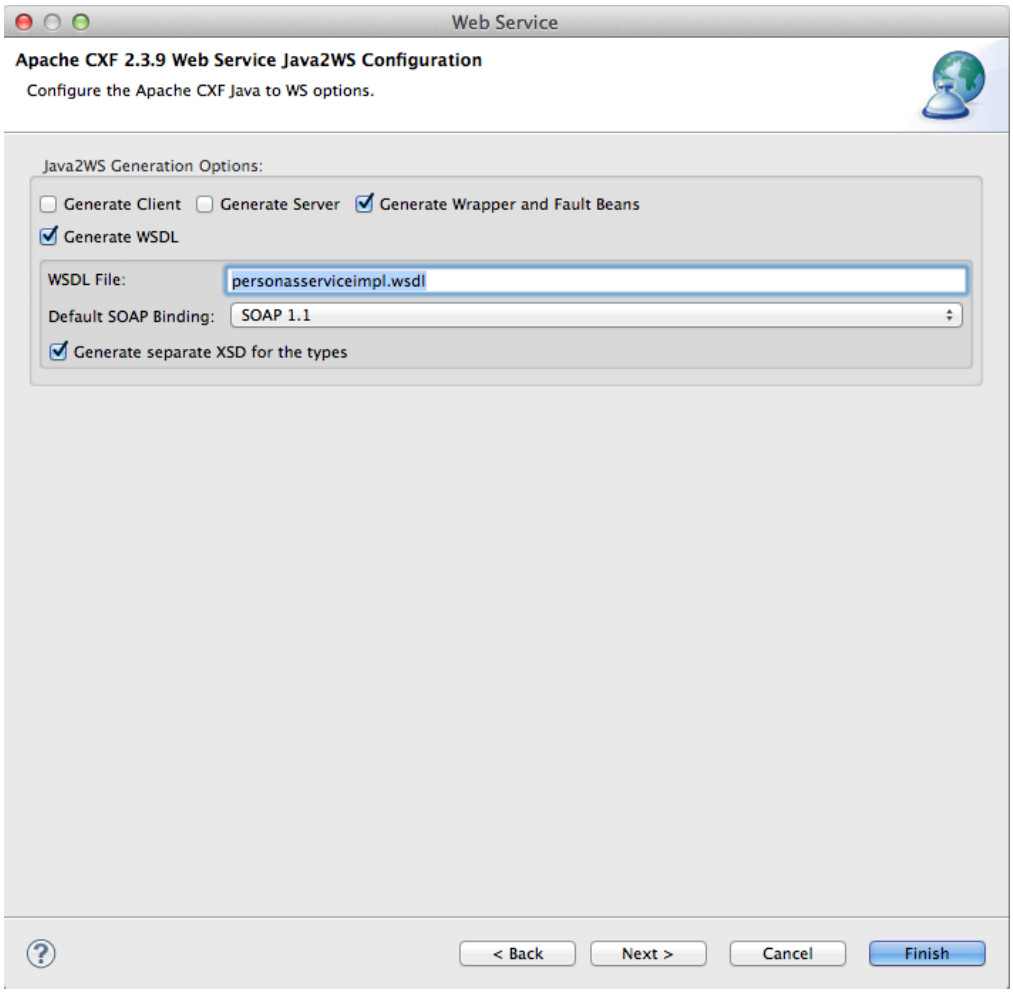


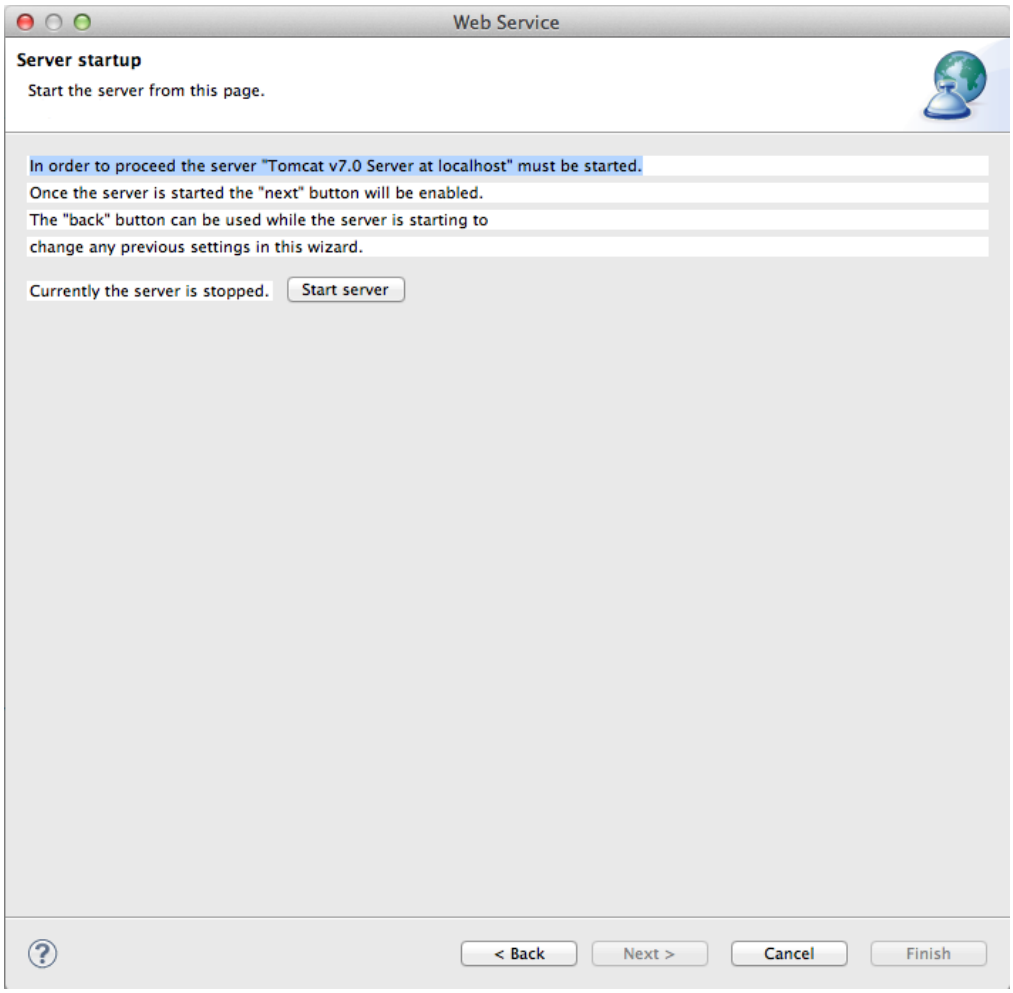

Service Deployment Configuration
Service Deployment Configuration
Choose from the list of runtimes and deployment servers, or use the default settings.
Server-Side Deployment Selection:
Choose server first
Choose Web service runtime first
Explore options
Web service runtime:
Apache Axis
Apache Axis2
Apache CXF 2.x
Server runtime:
Existing Servers
Tomcat v7.0 Server at localhost
Server Types
OK
Cancel

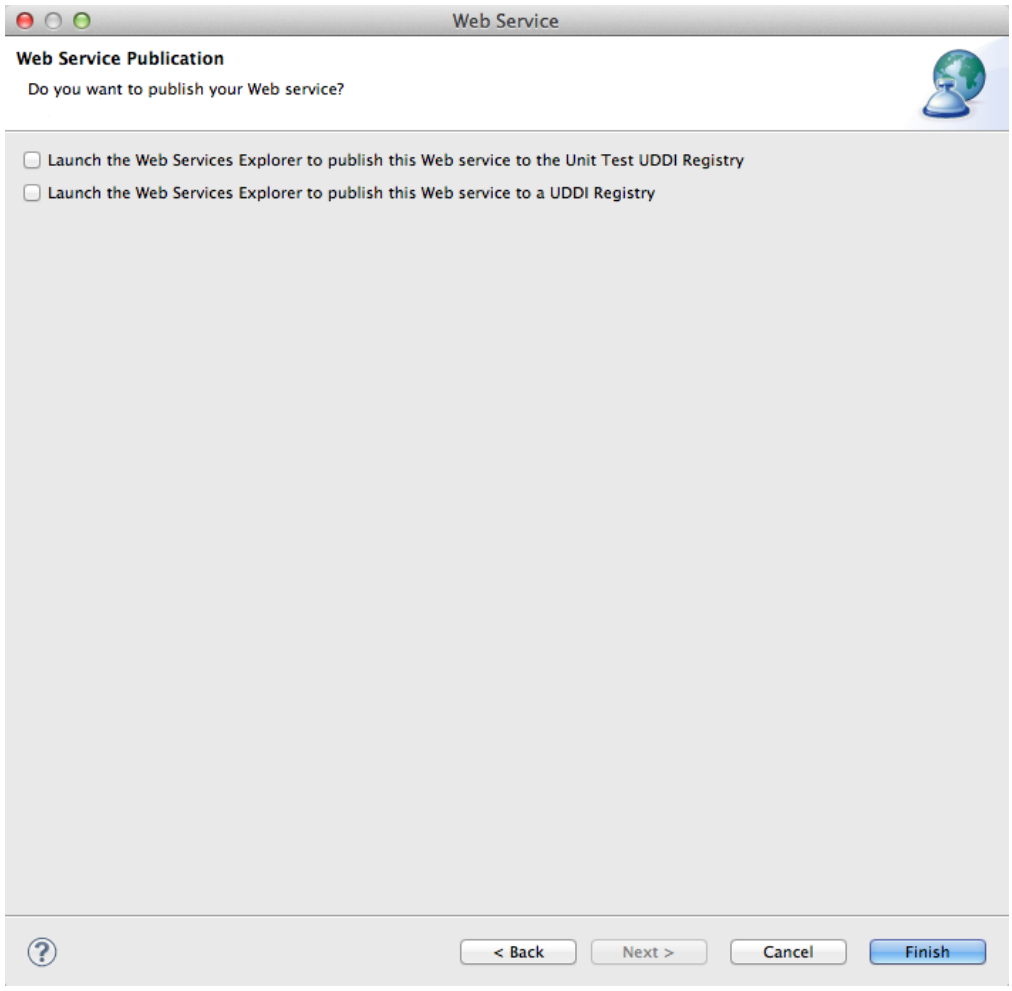

Specify Service Project Settings
Service project:
PersonasWebService
Service Project type:
Dynamic Web Project
Service EAR project:
Cancel
OK


Web Service
Web Services
Select a service implementation or definition and move the sliders to set the level of service and client generation.
Web service type:
Bottom up Java bean Web Service
Service implementation:
com.autentia.cxf.PersonasServiceImpl
Browse...
Start service
Configuration:
Server runtime: Tomcat v7.0 Server
Web service runtime: Apache CXF 2.x
Service project: PersonasWebService
Client type:
Java Proxy
No client
Configuration: No client generation.
Publish the Web service
Monitor the Web service
Do not show me this dialog box again.
< Back
Next >
Cancel
Finish


Web Service
Apache CXF 2.3.9 Web Service JAX-WS Annotations Configuration
Optionally, select the JAX-WS Annotations to add.
Select the Methods to annotate:
@WebMethod
@WebParam
@RequestWrapper
@ResponseWrapp
PersonasServiceImpl
addPersona(Persona)
getPersona(String)
getPersonas()
removePersona(String)
Preview:
/**
*
*/
package com.autentia.cxf;
import java.util.ArrayList;
import java.util.List;
import javax.jws.WebService;
/**
* @author cleon
*
< Back
Next >
Cancel
Finish


Web Service
Apache CXF 2.3.9 Web Service Java2WS Configuration
Configure the Apache CXF Java to WS options.
Java2WS Generation Options:
Generate Client
Generate Server
Generate Wrapper and Fault Beans
Generate WSDL
WSDL File:
personasserviceimpl.wsdl
Default SOAP Binding:
SOAP 1.1
Generate separate XSD for the types
< Back
Next >
Cancel
Finish

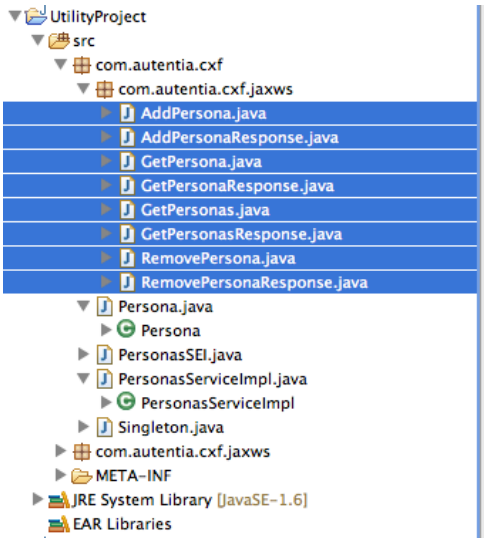




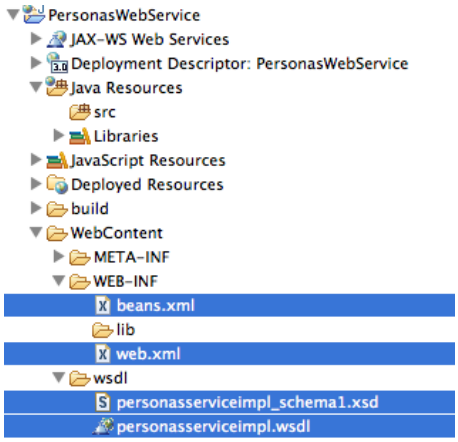
Lo primero que vemos es que nos ha anotado la clase que implementaba el servicio con las anotaciones de JAX-WS, y adicionalmente se han creado los tipos de request-response para cada una de las operaciones:

```
1  package com.autentia.cxf;
2
3  import java.util.ArrayList;
```

```java
import java.util.List;

import javax.jws.WebService;

/**
 * @author cleon
 *
 */
@WebService(targetNamespace = "http://cxf.autentia.com/", portName = "PersonasServiceImplPort"
public class PersonasServiceImpl implements PersonasSEI {

    /* (non-Javadoc)
     * @see com.autentia.cxf.PersonaSEI#addPersona(com.autentia.cxf.Persona)
     */
    @Override
    public void addPersona(Persona persona) {
        Singleton.getInstance().getPersonas().put(persona.getNombre(), persona);
    }

    /* (non-Javadoc)
     * @see com.autentia.cxf.PersonaSEI#getPersona(java.lang.String)
     */
    @Override
    public Persona getPersona(String nombre) throws Exception {
        Persona persona = Singleton.getInstance().getPersonas().get(nombre);
        if (persona==null){
            throw new Exception("La persona buscada no existe");
        }
        return persona;
    }

    /* (non-Javadoc)
     * @see com.autentia.cxf.PersonaSEI#getPersonas()
     */
    @Override
    public List<Persona> getPersonas() {
        return new ArrayList<Persona>(Singleton.getInstance().getPersonas().values());
    }

    /* (non-Javadoc)
     * @see com.autentia.cxf.PersonaSEI#removePersona(java.lang.String)
     */
    @Override
    public void removePersona(String nombre) throws Exception {
        Persona persona = Singleton.getInstance().getPersonas().get(nombre);
        if (persona==null){
            throw new Exception("La persona buscada no existe");
        }else{
            Singleton.getInstance().getPersonas().remove(nombre);
        }

    }

}
```



Despues se ha creado un proyecto web en el que lo mas relevante es que se ha generado el wsdl y adicionalmente se ha creado un xsd que contiene la definición de los tipos. Además se ha configurado el contenedor web y la configuración de Spring:



personasserviceimpl_schema1.xsd

```xml
<?xml version="1.0" encoding="utf-8"?><xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" ?
  <xs:element name="addPersona" type="tns:addPersona"/>
  <xs:element name="addPersonaResponse" type="tns:addPersonaResponse"/>
  <xs:element name="getPersona" type="tns:getPersona"/>
  <xs:element name="getPersonaResponse" type="tns:getPersonaResponse"/>
  <xs:element name="getPersonas" type="tns:getPersonas"/>
  <xs:element name="getPersonasResponse" type="tns:getPersonasResponse"/>
  <xs:element name="removePersona" type="tns:removePersona"/>
  <xs:element name="removePersonaResponse" type="tns:removePersonaResponse"/>
  <xs:complexType name="addPersona">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="tns:persona"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="persona">
    <xs:sequence>
      <xs:element minOccurs="0" name="apellidos" type="xs:string"/>
      <xs:element minOccurs="0" name="nombre" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="addPersonaResponse">
    <xs:sequence/>
  </xs:complexType>
  <xs:complexType name="getPersonas">
    <xs:sequence/>
  </xs:complexType>
  <xs:complexType name="getPersonasResponse">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="return" type="tns:persona"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="removePersona">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="removePersonaResponse">
    <xs:sequence/>
  </xs:complexType>
  <xs:complexType name="getPersona">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="getPersonaResponse">
    <xs:sequence>
      <xs:element minOccurs="0" name="return" type="tns:persona"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Exception" type="tns:Exception"/>
  <xs:complexType name="Exception">
    <xs:sequence>
      <xs:element minOccurs="0" name="message" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

personasserviceimpl.wsdl

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="PersonasServiceImplService" targetNamespace="http://cxf.autentia.com/
  <wsdl:types>
    <schema xmlns="http://www.w3.org/2001/XMLSchema">
<import namespace="http://cxf.autentia.com/" schemaLocation="personasserviceimpl_schema1.xsd"
</schema>
  </wsdl:types>
  <wsdl:message name="addPersonaResponse">
    <wsdl:part name="parameters" element="tns:addPersonaResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getPersonas">
    <wsdl:part name="parameters" element="tns:getPersonas">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="addPersona">
    <wsdl:part name="parameters" element="tns:addPersona">
    </wsdl:part>
  </wsdl:message>
```

```
  <wsdl:message name="getPersonasResponse">
    <wsdl:part name="parameters" element="tns:getPersonasResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="removePersonaResponse">
    <wsdl:part name="parameters" element="tns:removePersonaResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="removePersona">
    <wsdl:part name="parameters" element="tns:removePersona">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getPersonaResponse">
    <wsdl:part name="parameters" element="tns:getPersonaResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getPersona">
    <wsdl:part name="parameters" element="tns:getPersona">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="Exception">
    <wsdl:part name="Exception" element="tns:Exception">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="PersonasServiceImpl">
    <wsdl:operation name="addPersona">
      <wsdl:input name="addPersona" message="tns:addPersona">
    </wsdl:input>
      <wsdl:output name="addPersonaResponse" message="tns:addPersonaResponse">
    </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getPersonas">
      <wsdl:input name="getPersonas" message="tns:getPersonas">
    </wsdl:input>
      <wsdl:output name="getPersonasResponse" message="tns:getPersonasResponse">
    </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="removePersona">
      <wsdl:input name="removePersona" message="tns:removePersona">
    </wsdl:input>
      <wsdl:output name="removePersonaResponse" message="tns:removePersonaResponse">
    </wsdl:output>
      <wsdl:fault name="Exception" message="tns:Exception">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getPersona">
      <wsdl:input name="getPersona" message="tns:getPersona">
    </wsdl:input>
      <wsdl:output name="getPersonaResponse" message="tns:getPersonaResponse">
    </wsdl:output>
      <wsdl:fault name="Exception" message="tns:Exception">
    </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="PersonasServiceImplServiceSoapBinding" type="tns:PersonasServiceImpl">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="addPersona">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="addPersona">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="addPersonaResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getPersonas">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="getPersonas">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getPersonasResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getPersona">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="getPersona">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getPersonaResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="Exception">
        <soap:fault name="Exception" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="removePersona">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="removePersona">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="removePersonaResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="Exception">
        <soap:fault name="Exception" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="PersonasServiceImplService">
    <wsdl:port name="PersonasServiceImplPort" binding="tns:PersonasServiceImplServiceSoapBind
      <soap:address location="http://localhost:8080/PersonasWebService/services/PersonasServi
    </wsdl:port>
  </wsdl:service>
```

```
</wsdl:definitions>
```

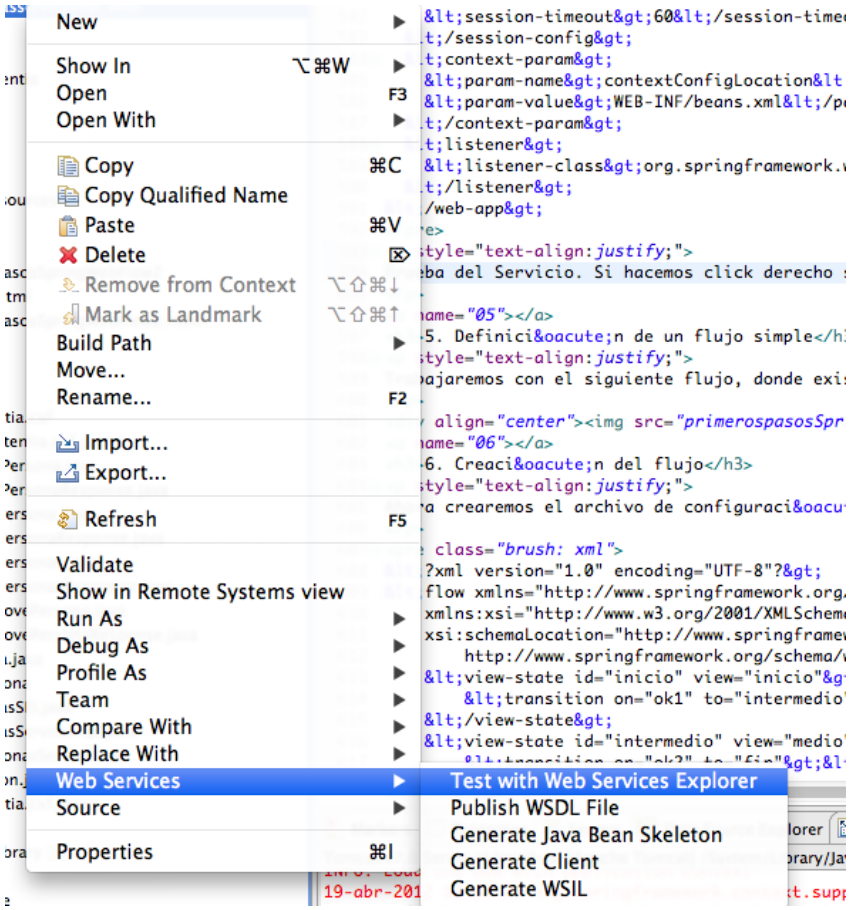
beans.xml

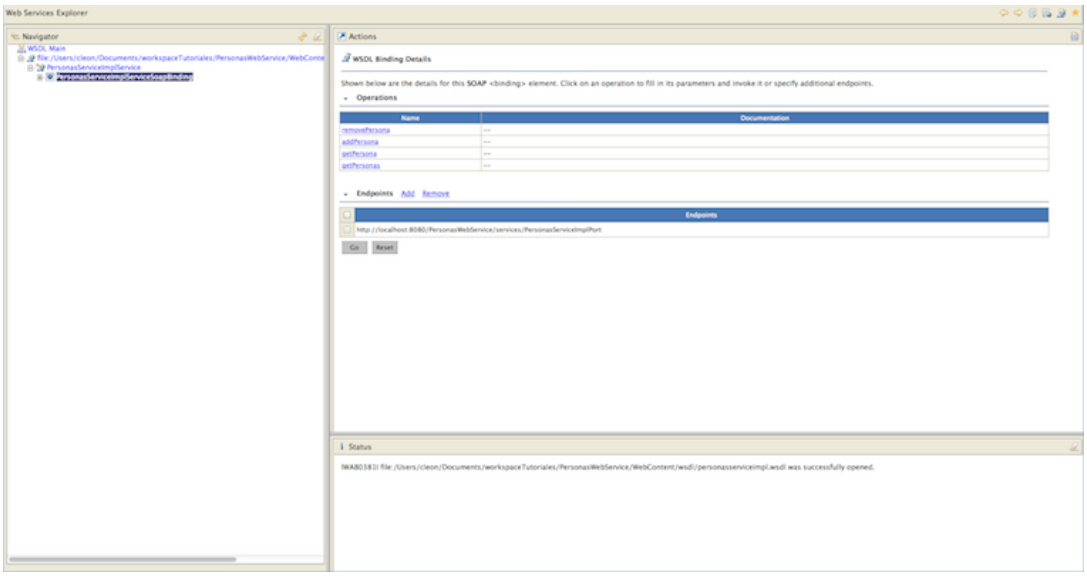
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:jaxws="http://cxf.apache.org/j
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework
    <import 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" />
    <jaxws:endpoint xmlns:tns="http://cxf.autentia.com/" id="personasservice"
        implementor="com.autentia.cxf.PersonasServiceImpl" wsdlLocation="wsdl/personasservicei
        endpointName="tns:PersonasServiceImplPort" serviceName="tns:PersonasServiceImplService
        address="/PersonasServiceImplPort">
        <jaxws:features>
            <bean class="org.apache.cxf.feature.LoggingFeature" />
        </jaxws:features>
    </jaxws:endpoint>
</beans>
```

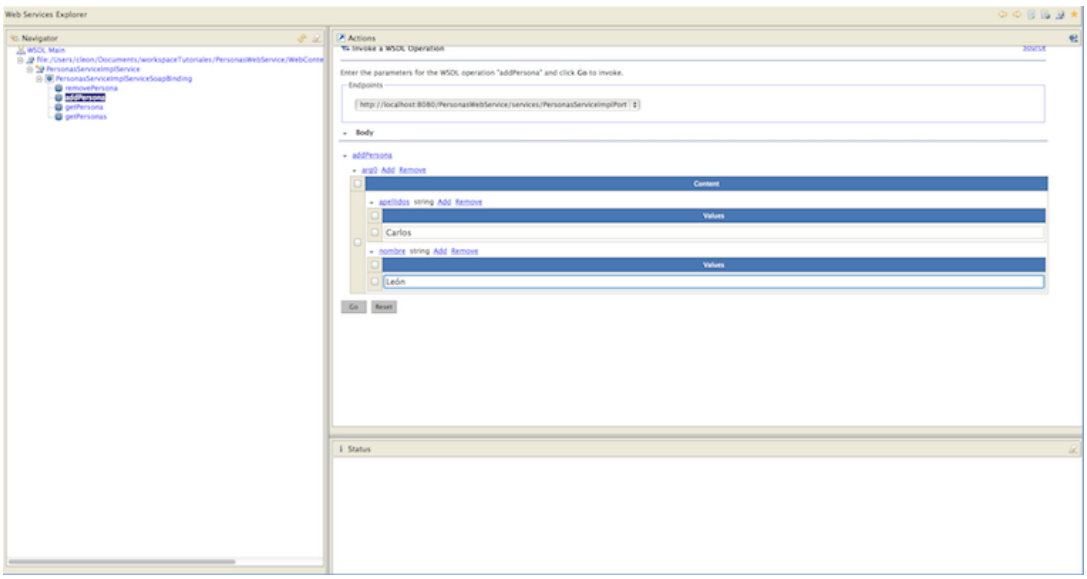
web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/
  <display-name>PersonasWebService</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <description>Apache CXF Endpoint</description>
    <display-name>cxf</display-name>
    <servlet-name>cxf</servlet-name>
    <servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>cxf</servlet-name>
    <url-pattern>/services/*</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>60</session-timeout>
  </session-config>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>WEB-INF/beans.xml</param-value>
  </context-param>
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
</web-app>
```

Prueba del Servicio. Si hacemos click derecho sobre el wsdl seleccionando webservices-->Test with webservices explorer y realizamos una petición de addPersona







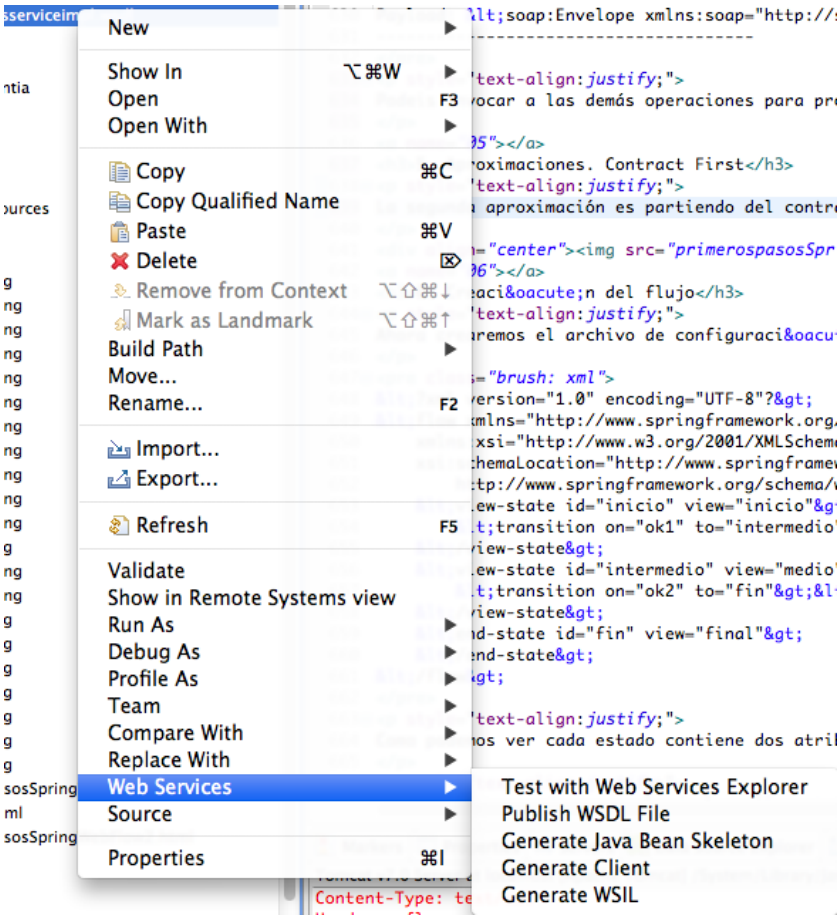
Si revisamos los logs del servidor, podemos ver todos los datos de la petición y la respuesta

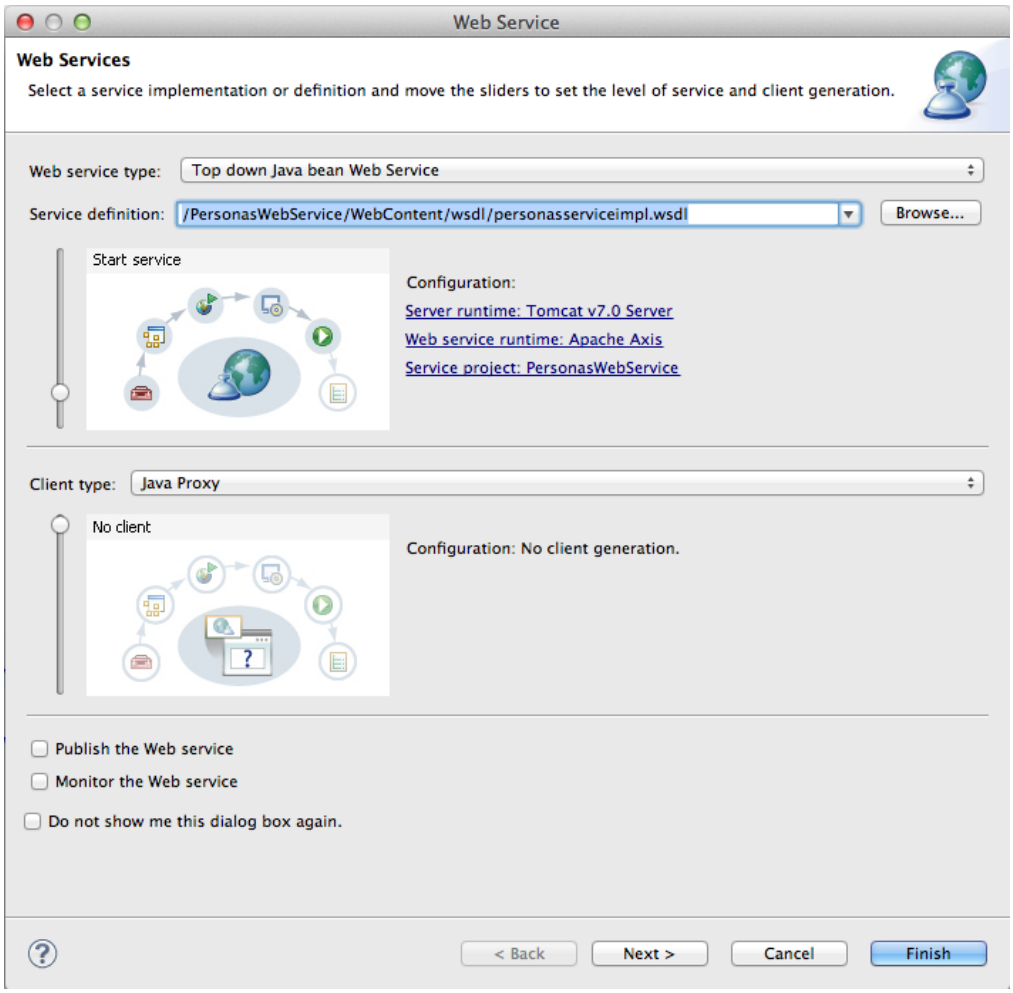
```
INFO: Inbound Message
----------------------------
ID: 1
Address: http://localhost:8080/PersonasWebService/services/PersonasServiceImplPort
Encoding: UTF-8
Http-Method: POST
Content-Type: text/xml; charset=utf-8
Headers: {Accept=[application/soap+xml, application/dime, multipart/related, text/*], cache-co
Payload: <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:q0=
  <soapenv:Body>
    <q0:addPersona>
      <arg0>
        <apellidos>Carlos</apellidos>
        <nombre>León</nombre>
      </arg0>
    </q0:addPersona>
  </soapenv:Body>
</soapenv:Envelope>

--------------------------------------
19-abr-2012 13:30:23 org.apache.cxf.interceptor.AbstractLoggingInterceptor log
INFO: Outbound Message
----------------------------
ID: 1
Encoding: UTF-8
Content-Type: text/xml
Headers: {}
Payload: <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soap:Body><ns2
--------------------------------------
```

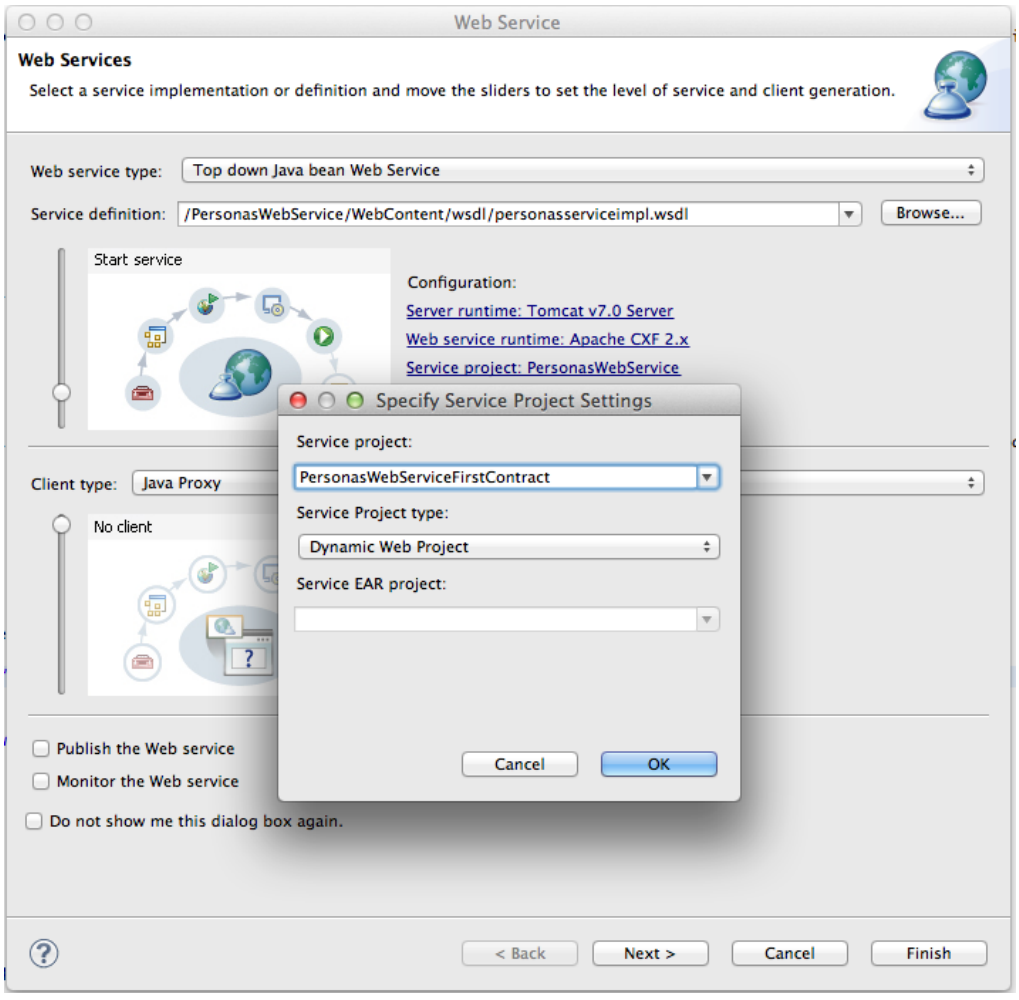
Podeis invocar a las demás operaciones para probar que todo está ok.

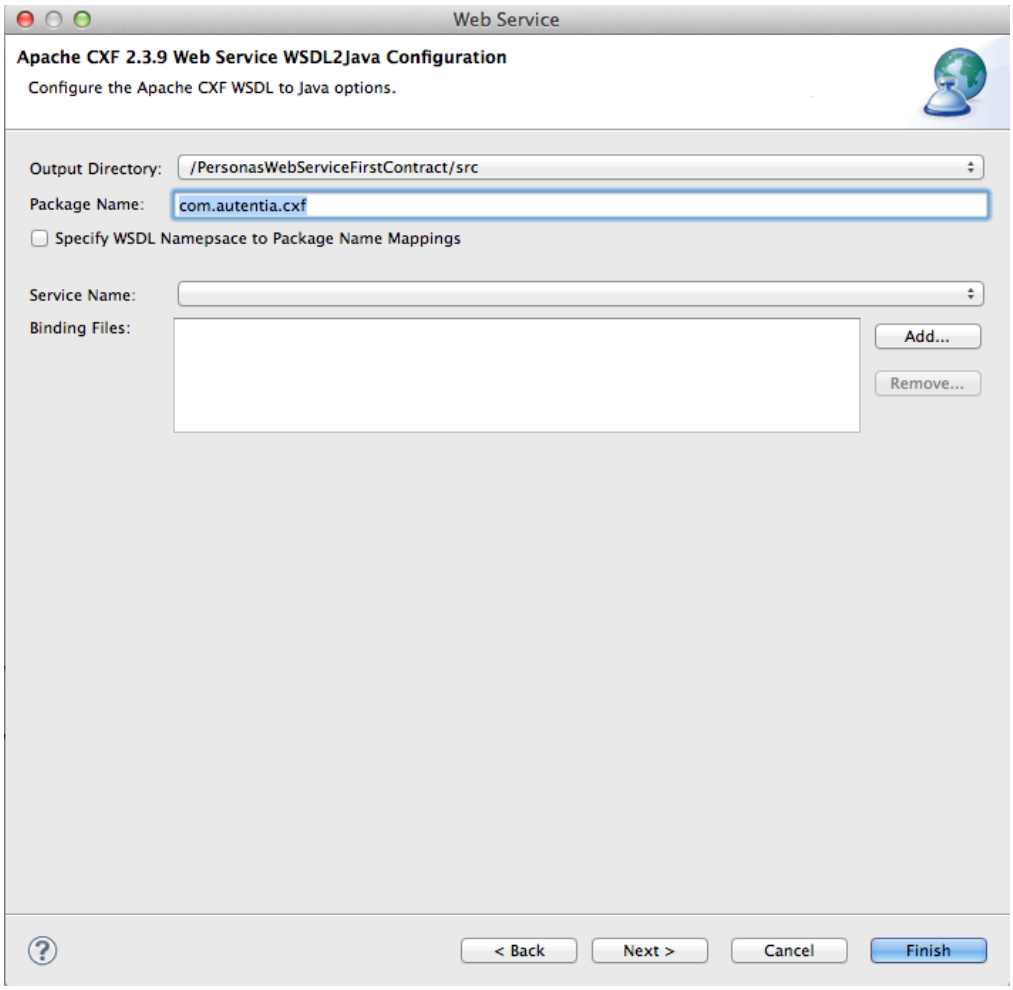
5. Aproximaciones. Contract First

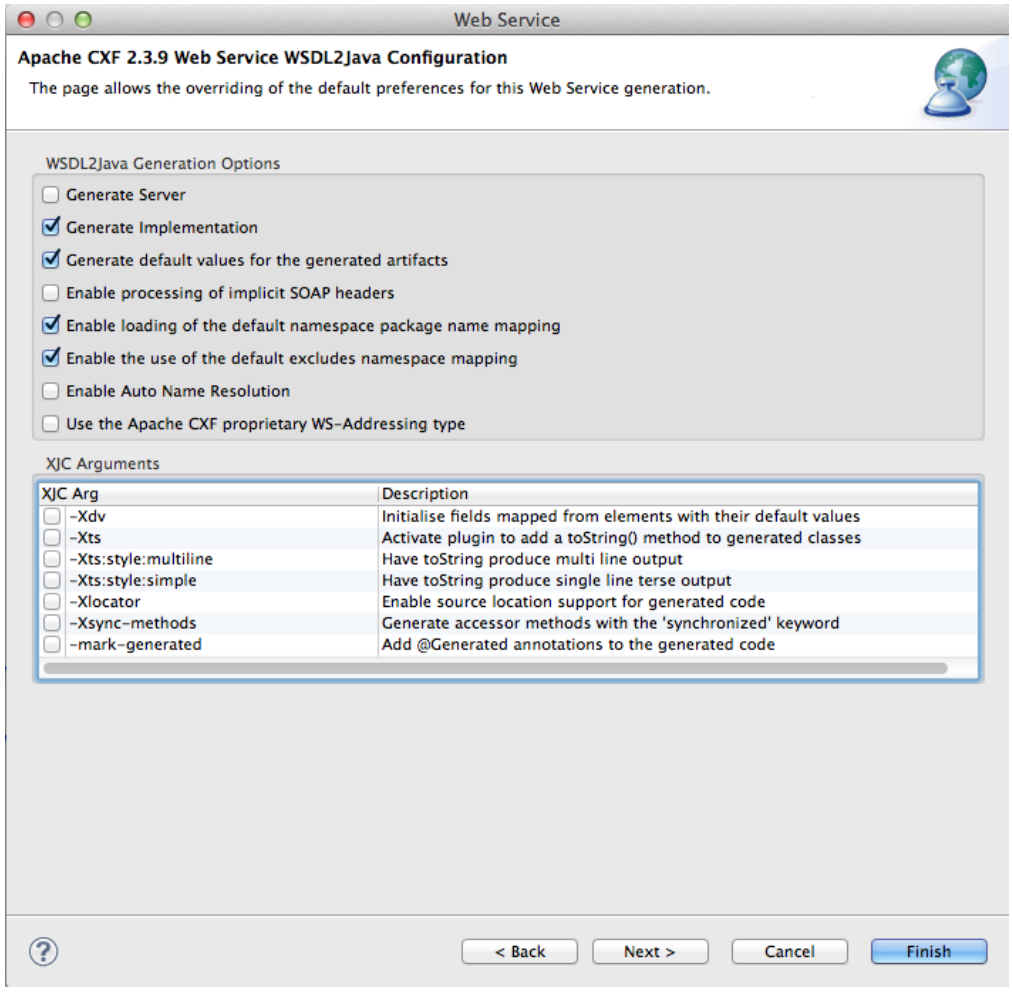
La segunda aproximación es partiendo del contrato, es decir, el wsdl. Para ello utilizaremos el wsdl generado en el punto anterior. Si realizamos un click derecho sobre el WSDL y después realizamos los siguientes pasos:

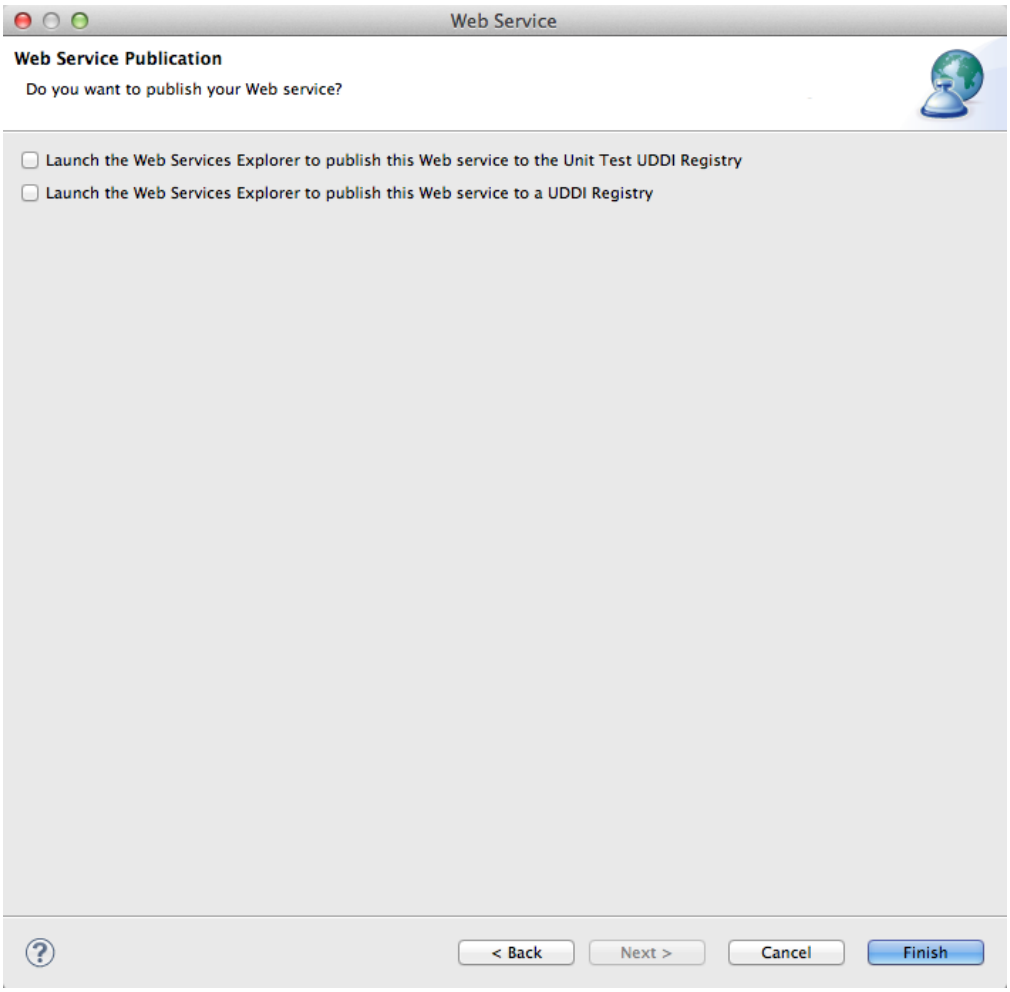



Web Service
Web Services
Select a service implementation or definition and move the sliders to set the level of service and client generation.
Web service type:
Top down Java bean Web Service
Service definition:
/PersonasWebService/WebContent/wsdl/personasserviceimpl.wsdl
Browse...
Start service
Configuration:
Server runtime: Tomcat v7.0 Server
Web service runtime: Apache Axis
Service project: PersonasWebService
Client type:
Java Proxy
No client
Configuration: No client generation.
Publish the Web service
Monitor the Web service
Do not show me this dialog box again.
< Back
Next >
Cancel
Finish

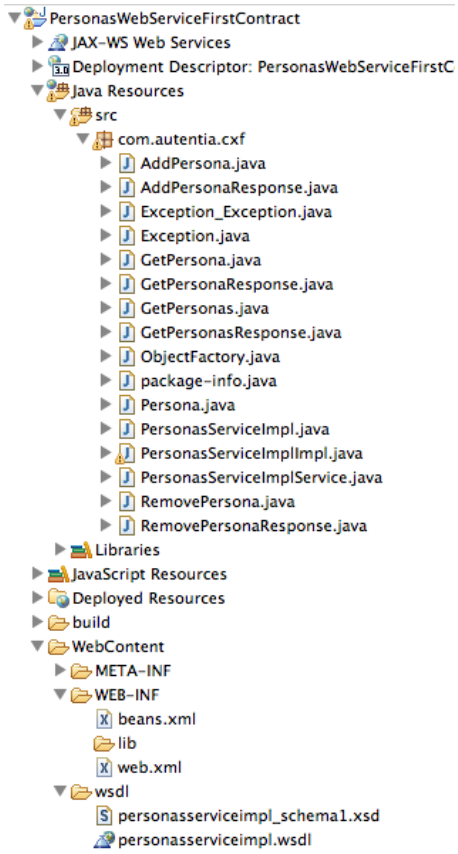

Web Service
Web Services
Select a service implementation or definition and move the sliders to set the level of service and client generation.
Web service type:
Top down Java bean Web Service
Service definition:
/PersonasWebService/WebContent/wsdl/personasserviceimpl.wsdl
Browse...
Start service
Configuration:
Server runtime: Tomcat v7.0 Server
Web service runtime: Apache CXF 2.x
Service project: PersonasWebService
Specify Service Project Settings
Service project:
PersonasWebServiceFirstContract
Service Project type:
Dynamic Web Project
Service EAR project:
Cancel
OK
Client type:
Java Proxy
No client
Publish the Web service
Monitor the Web service
Do not show me this dialog box again.
< Back
Next >
Cancel
Finish


Web Service
Apache CXF 2.3.9 Web Service WSDL2Java Configuration
Configure the Apache CXF WSDL to Java options.
Output Directory:
/PersonasWebServiceFirstContract/src
Package Name:
com.autentia.cxf
Specify WSDL Namepsace to Package Name Mappings
Service Name:
Binding Files:
Add...
Remove...
< Back
Next >
Cancel
Finish


Web Service
Apache CXF 2.3.9 Web Service WSDL2Java Configuration
The page allows the overriding of the default preferences for this Web Service generation.
WSDL2Java Generation Options
Generate Server
Generate Implementation
Generate default values for the generated artifacts
Enable processing of implicit SOAP headers
Enable loading of the default namespace package name mapping
Enable the use of the default excludes namespace mapping
Enable Auto Name Resolution
Use the Apache CXF proprietary WS-Addressing type
XJC Arguments
XJC Arg
Description
-Xdv
Initialise fields mapped from elements with their default values
-Xts
Activate plugin to add a toString() method to generated classes
-Xts:style:multiline
Have toString produce multi line output
-Xts:style:simple
Have toString produce single line terse output
-Xlocator
Enable source location support for generated code
-Xsync-methods
Generate accessor methods with the 'synchronized' keyword
-mark-generated
Add @Generated annotations to the generated code
< Back
Next >
Cancel
Finish



Generaremos el esqueleto del webservice, en el cual tan solo deberemos modificar la clase PersonasServiceImplImpl con nuestra implementación, ya que al partir del contrato, no tenemos la lógica de negocio del servicio.



6. Conclusiones

Como hemos podido comprobar, es bastante sencillo realizar webservices con estas herramientas, ya que las WST nos proveen una forma cómoda de trabajar con los comandos wsdl2java y java2wsdl que implementa la distribución de Apache CXF. Aunque es una forma muy rápida de implementar los webservices, os recomiendo que lo hagais simplemente para construir el esqueleto de vuestro servicio para después pasarlo a un proyecto maven.

Los siguientes tutoriales sobre apache cxf serán construidos con maven basandonos en la arquitectura que hemos visto.

Como nota significativa y adicional a la generación del webservice desde una clase java, es que la implementación del WSDL es siempre document-literal, ya que los webservices son mucho mas interoperables trabajando en este formato (revisar http://www.ibm.com/developerworks/webservices/library/ws-whichwsdl/). Si partimos de la generación del webservice desde el WSDL y el formato es rpc-encoded, apache cxf obtendremos un error (ya que no cumple la WS-I Compliance, mas información en http://www.ws-i.org/), ya que se está quedando obsoleto el formato RPC. Para terminar, recomiendo el uso de apache cxf para implementar WebServices, dado que cubre casi todas las necesidades y está bastante documentado, además de ser motor de webservices por defecto en muchas plataformas ESB opensource y de pago.

Cualquier duda o sugerencia podeis comentarlo.

Saludos.

A continuación puedes evaluarlo:

Regístrate para evaluarlo

Por favor, vota +1 o compártelo si te pareció interesante

Share | 0

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

» Registrate y accede a esta y otras ventajas **«**

Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5

IMPULSA Impulsores Comunidad ¿Ayuda?

---- sin clicks

0 personas han traído clicks a esta página

+ + + + + + + +

powered by karmacracy

Copyright 2003-2012 © All Rights Reserved | Texto legal y condiciones de uso | Banners | Powered by Autentia | Contacto

XHTML 1.0 CSS RSS ATOM