CREAR UN SERVICIO WEB BASICO CON JAVA AXIS2.

Víctor J. Sosa
vjsosa@tamps.cinvestav.mx

En este documento explicaré brevemente cómo construir un servicio web con Java **Axis2** y cómo invocarlo desde un cliente o directamente de un navegador. El ejemplo se desarrolla utilizando el IDE **Eclipse** y el contenedor de servlets **Tomcat** en un entorno **Windows**. La lógica del servicio será muy simple, ya que sólo deberá decir "Hola Mundo" o cualquier otra cadena de caracteres que queramos.

Requisitos para empezar.

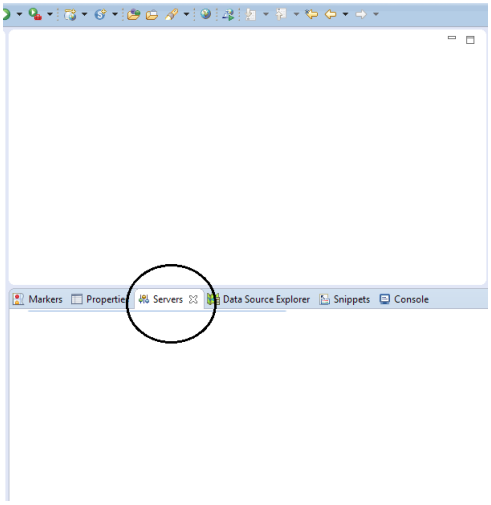
Eclipse:
Para el desarrollo del ejemplo se utilizó como IDE a Eclipse. En caso de no contar con Eclipse, el primer paso será descargarlo. La versión de Eclipse que se utilizó fue **Eclipse IDE for Java EE Developers** (**Luna**), el cual se puede descargar de: https://eclipse.org/downloads/index-developer.php, tomar en cuenta que se está utilizando Java EE, que incluye las librerías necesarias para el desarrollo de aplicaciones web.

AXIS2:
Como motor de servicios web, en este ejemplo utilizamos Axis2, en su versión binaria 1.6.2 para Windows. Se puede descargar de:
http://axis.apache.org/axis2/java/core/download.cgi

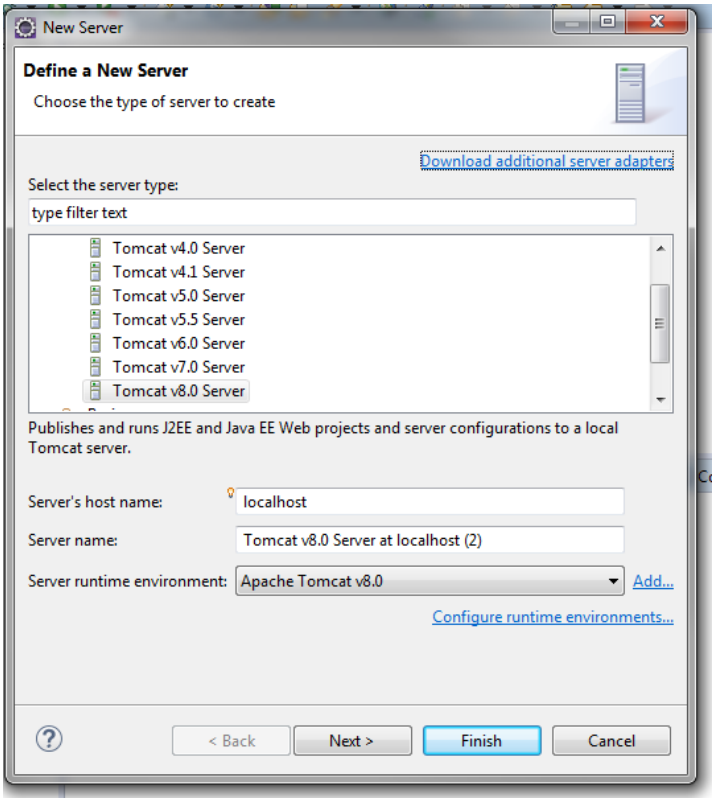
Tomcat:
También es necesario que instalemos el contenedor de servlets Tomcat. La versión utilizada para este ejemplo fue la 8.0, la cual se puede descargar de: http://tomcat.apache.org/download-80.cgi

Configuración de Tomcat en Eclipse.

Una vez que hemos descargado e instalado este software, abrimos eclipse y lo configuramos para su uso con Tomcat. Para hacer que eclipse reconozca nuestro Tomcat 8.0 nos iremos a:
Windows-> Show View-> Other-> Server -> Server
Al dar Ok ahí, aparecerá una pestaña que dice servers como se muestra en la figura



Nos posicionamos en esa ventana, oprimimos botón derecho y nos vamos a:
New-> New Server mostrando la siguiente ventana…

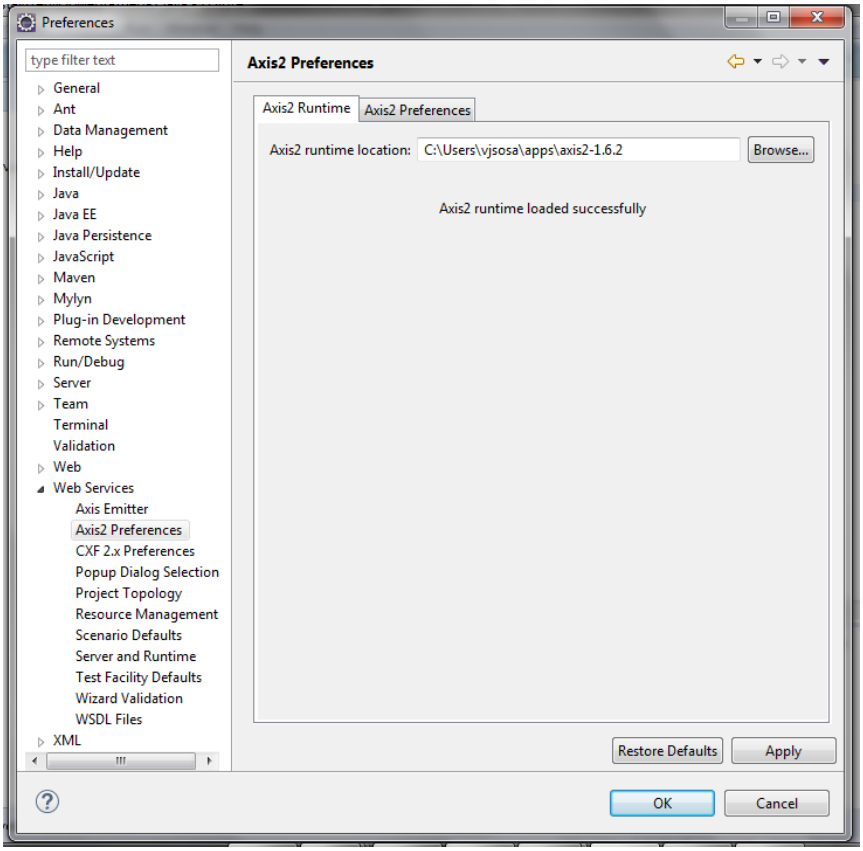


Ahí elegimos nuestro Tomcat v8.0 e indicamos el lugar donde lo hemos instalado (un wizard nos guiará durante la configuración). Después de que ya nos aparece que el servidor está incluido como parte de la configuración de Eclipse, podemos dar doble click donde indica el nombre del servidor: Tomcat v8.0 Server.
En la parte que dice “server location” vemos que por default aparece “use workspace metadata” lo que indica que usaremos una configuración de Tomcat que es para desarrollo y no para producción (es decir, no modificaremos directamente los archivos que tenemos instalados en nuestro Tomcat local). Esta opción es la que usaremos para el ejemplo. Sin embargo, podríamos también cambiar la parte que dice “server location” para que tenga la opción “use tomcat installation”, lo que haría que usemos directamente los archivos del Tomcat que tenemos instalado sin instanciar otro para uso exclusivo de este ejemplo. En este último caso podemos ver que usando el icono del mundo de Eclipse, “open web browser”, la página principal de Tomcat se encuentra en http://localhost:8080/. De lo contrario, que es el caso de nuestro ejemplo, en esa dirección no encontraremos la página principal de Tomcat. Para mayor información sobre como configurar Tomcat en Eclipse (utilizando un ejemplo con Tomcat 7) se puede consultar el siguiente tutorial https://www.youtube.com/watch?v=HUYD6QloND8

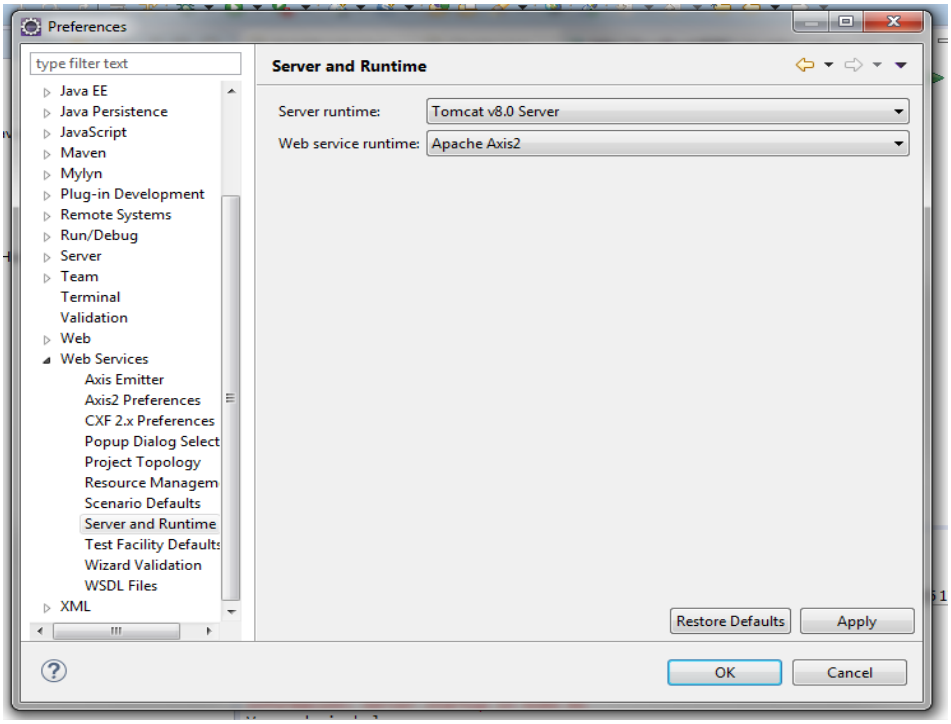
Configuración de Axis2 en Eclipse

Para configurar Axis2 en Eclipse nos vamos a:
Windows->Preferences->->Web Services->Axis2 Preferences

Ahí pondremos la ruta donde fue instalado Axis2 (distribución binaria), en mi caso:
C:\Users\vjsosa\apps\axis2-1.6.2
La siguiente imagen muestra este proceso.



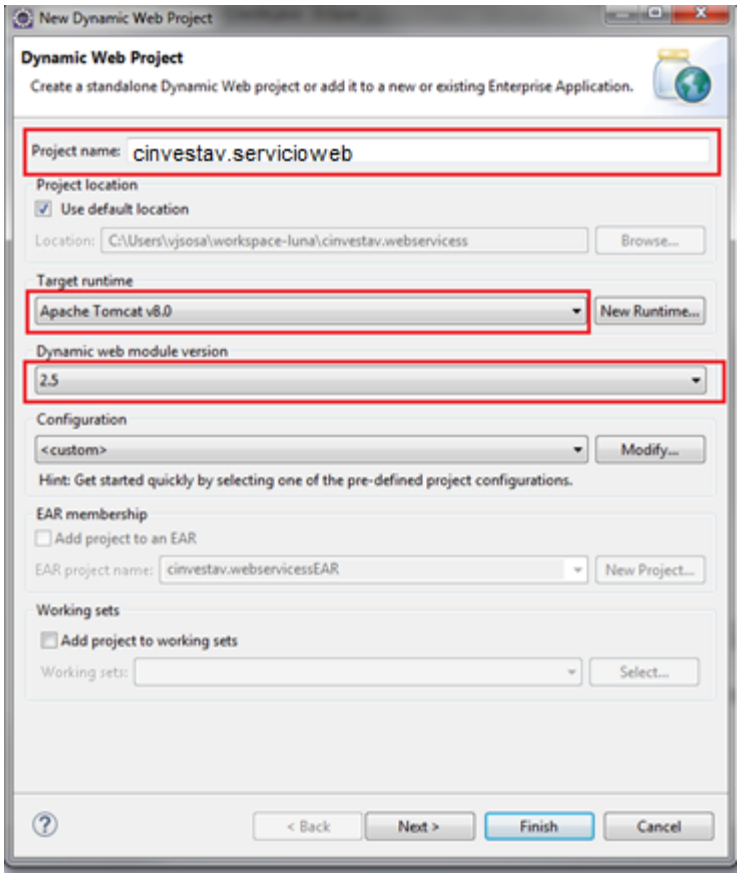
Sin movernos de la pestaña de Web Services, nos pasamos ahora a la opción de "Server and Runtime", accediendo desde la pestaña original en Eclipse, sería:
Windows->Preferences->Web Services->Server and Runtime.
En este lugar indicamos que usaremos Tomcat como Server Runtime y Axis2 como Webservice Runtime. La siguiente imagen muestra este proceso.

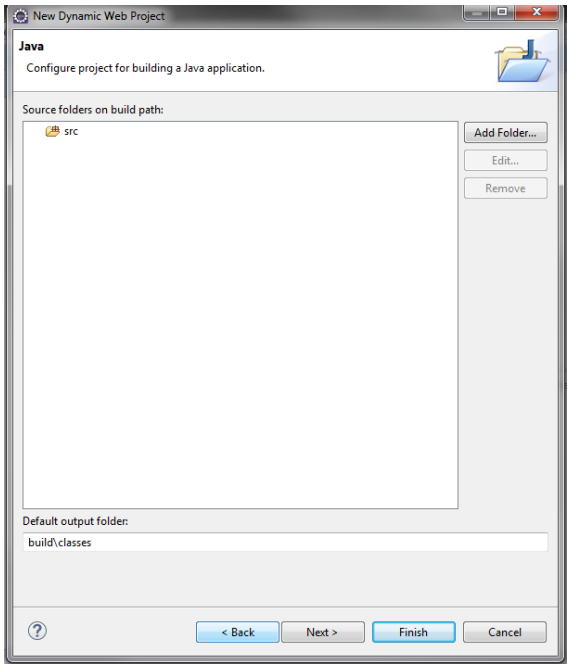


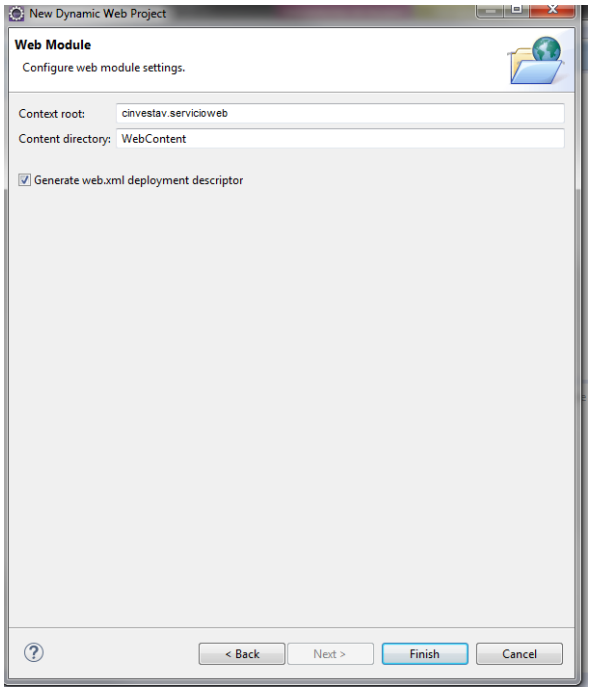
Creación de un Servicio Web

A continuación vamos a crear nuestro ejemplo de proyecto web en eclipse. Primero nos vamos a: File-> New-> Dynamic Web Project

Al proyecto (paquete) le pondremos cinvestav.servicioweb. Como nota adicional, quiero mencionar que dado que surgió un problema de compatibilidad entre la versión de Axis2 que descargué con la versión 3.1 del paquete Dynamic Web Module, fue necesario cambiar la versión del Dynamic Web Module a 2.5. Esta versión es compatible con la versión de Axis2 instalada. Las siguientes figuras muestran el proceso y los datos asignados.


New Dynamic Web Project
Dynamic Web Project
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.
Project name: cinvestav.servicioweb
Project location
Use default location
Location: C:\Users\vjsosa\workspace-luna\cinvestav.webservicess
Browse...
Target runtime
Apache Tomcat v8.0
New Runtime...
Dynamic web module version
2.5
Configuration
<custom>
Modify...
Hint: Get started quickly by selecting one of the pre-defined project configurations.
EAR membership
Add project to an EAR
EAR project name: cinvestav.webservicessEAR
New Project...
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Finish
Cancel

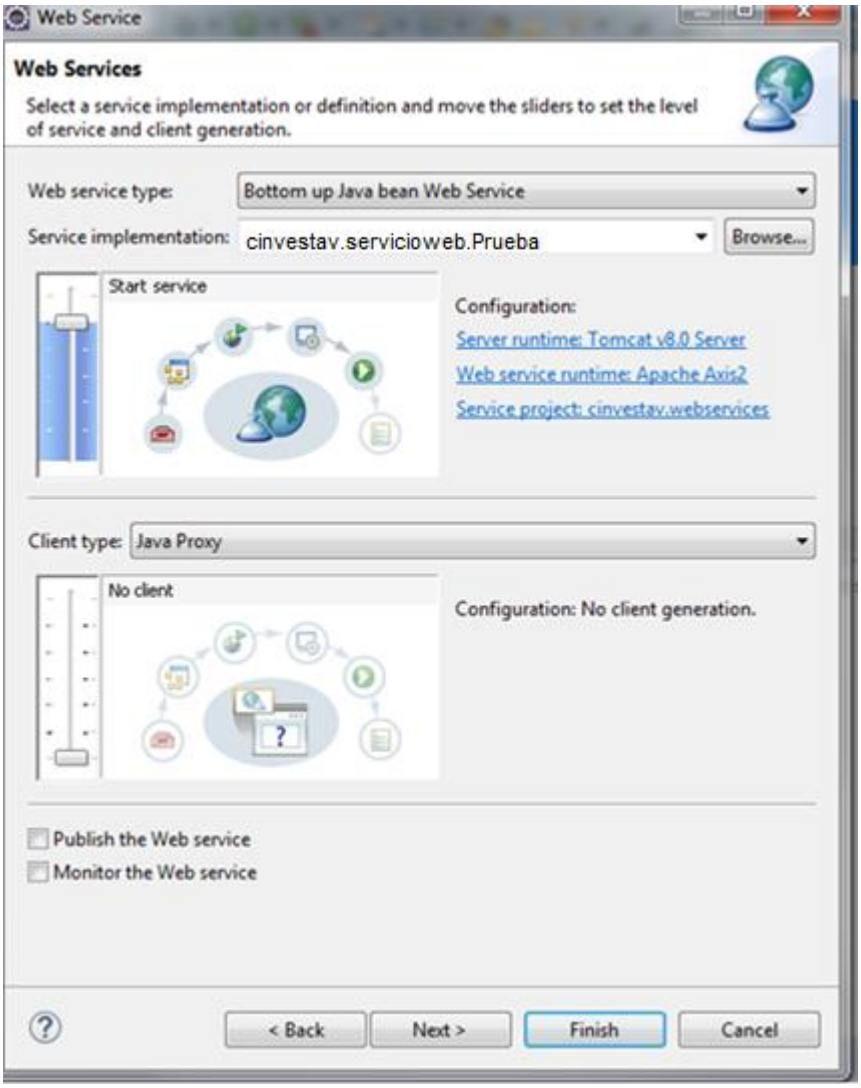

New Dynamic Web Project
Java
Configure project for building a Java application.
Source folders on build path:
src
Add Folder...
Edit...
Remove
Default output folder:
build\classes
< Back
Next >
Finish
Cancel


New Dynamic Web Project
Web Module
Configure web module settings.
Context root: cinvestav.servicioweb
Content directory: WebContent
Generate web.xml deployment descriptor
< Back
Next >
Finish
Cancel

Una vez que hemos creado el proyecto, procedemos a crear nuestro servicio web, mediante crear la clase Prueba. Esta clase se crea como cualquier otra en la sección de Java Resource | src.

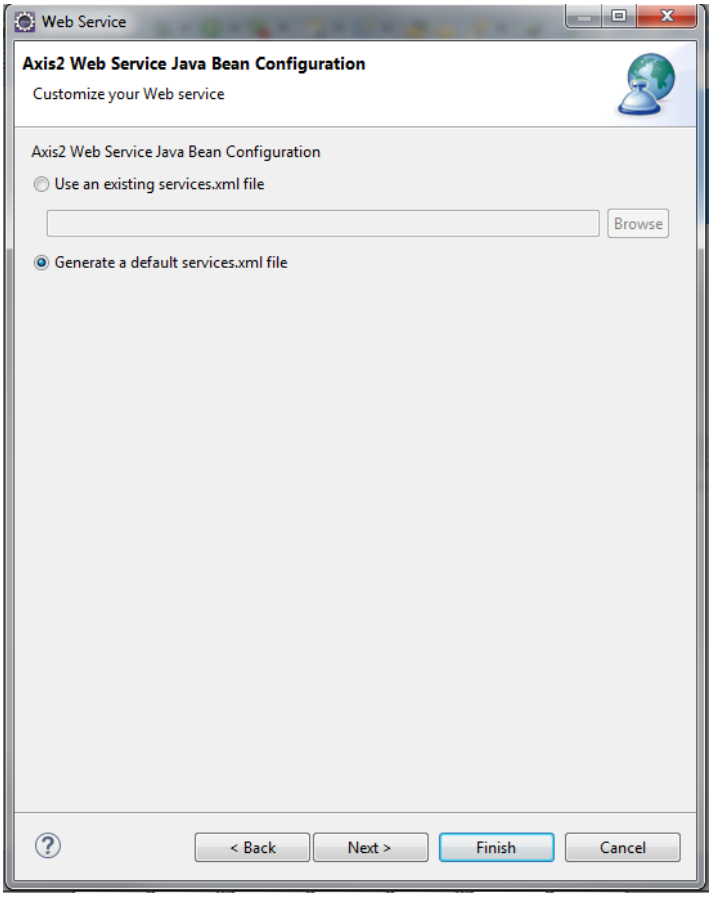
```
package cinvestav.servicioweb;

public class Prueba {
	public String sayHello(){
		System.out.println("Probando sayHello");
		return "Hola Mundo!!";
		}
}
```

A continuación es momento de registrar esta clase como servicio web. La forma de hacerlo es la siguiente: Posicionar el mouse sobrela clase y oprimir botón derecho y dar:
New-> Other-> Web Service-> Web Service



La imagen muestra la información de configuración que aparecerá. Es importante verificar que en configuración aparezcan Tomcat v8.0 (o la versión que se haya instalado) en Server Runtime y Apache Axis2 como Web Server Runtime.

El proceso “natural” de creación de los Web Services es “Top Down”, es decir, primero crear la descripción del servicio (WSDL) y a partir de ella la clase en cuestión, pero para este ejemplo se hizo al revés “Bottom up”, primero la clase y luego la descripción. Una vez hecho esto, seguimos todos los pasos del Wizard.



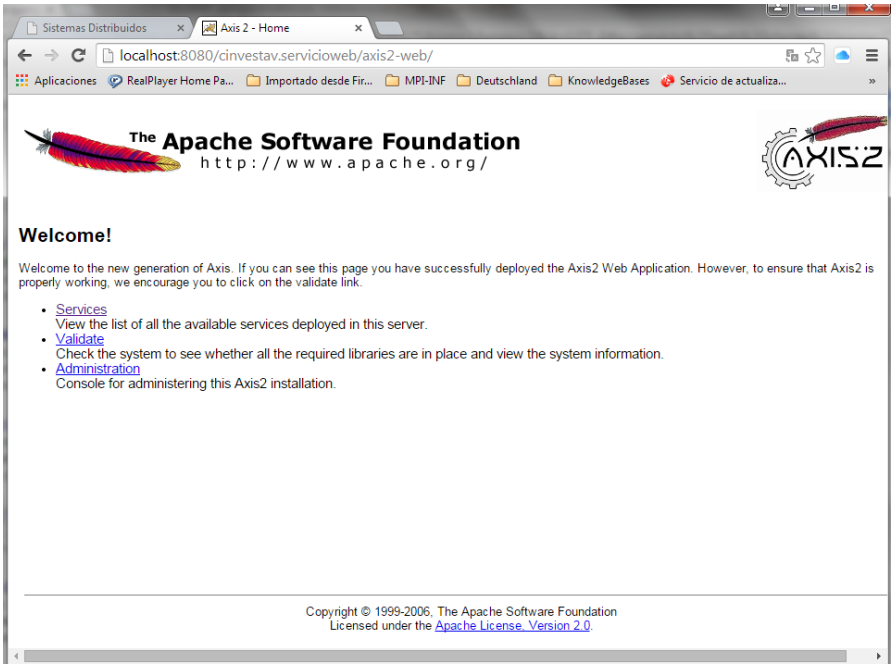
Veremos que en uno de ellos nos pedirá arrancar el servidor Tomcat (en caso de que no lo esté).

A continuación abrimos un navegador y ponemos la dirección:

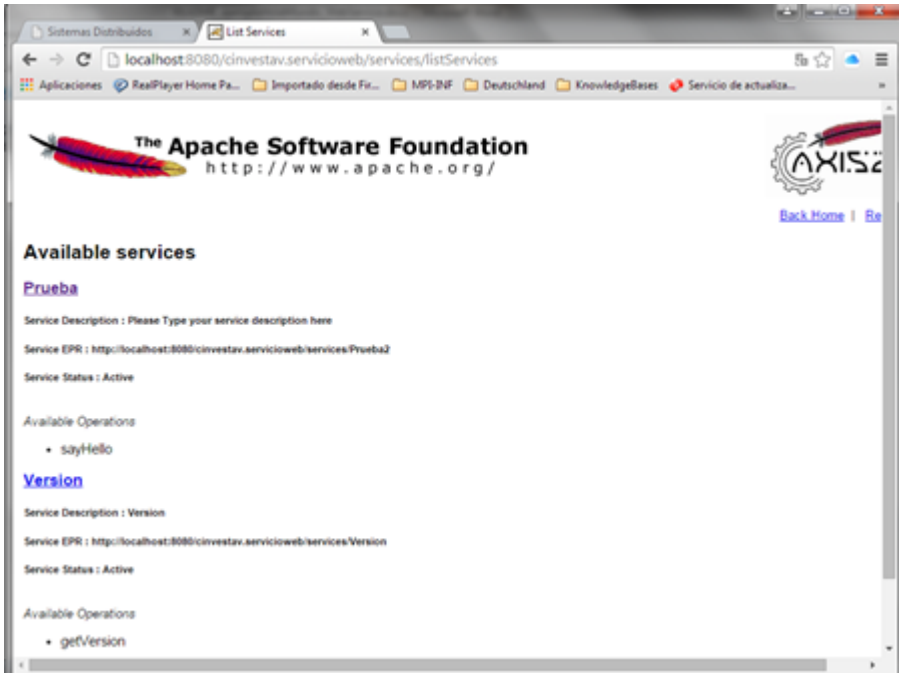
http://localhost:8080/cinvestav.servicioweb/axis2-web/

Si estuviéramos usamos la instalación directa del Tomcat (opción “use Tomcat installation”), la url sería: http://localhost:8080/axis2/

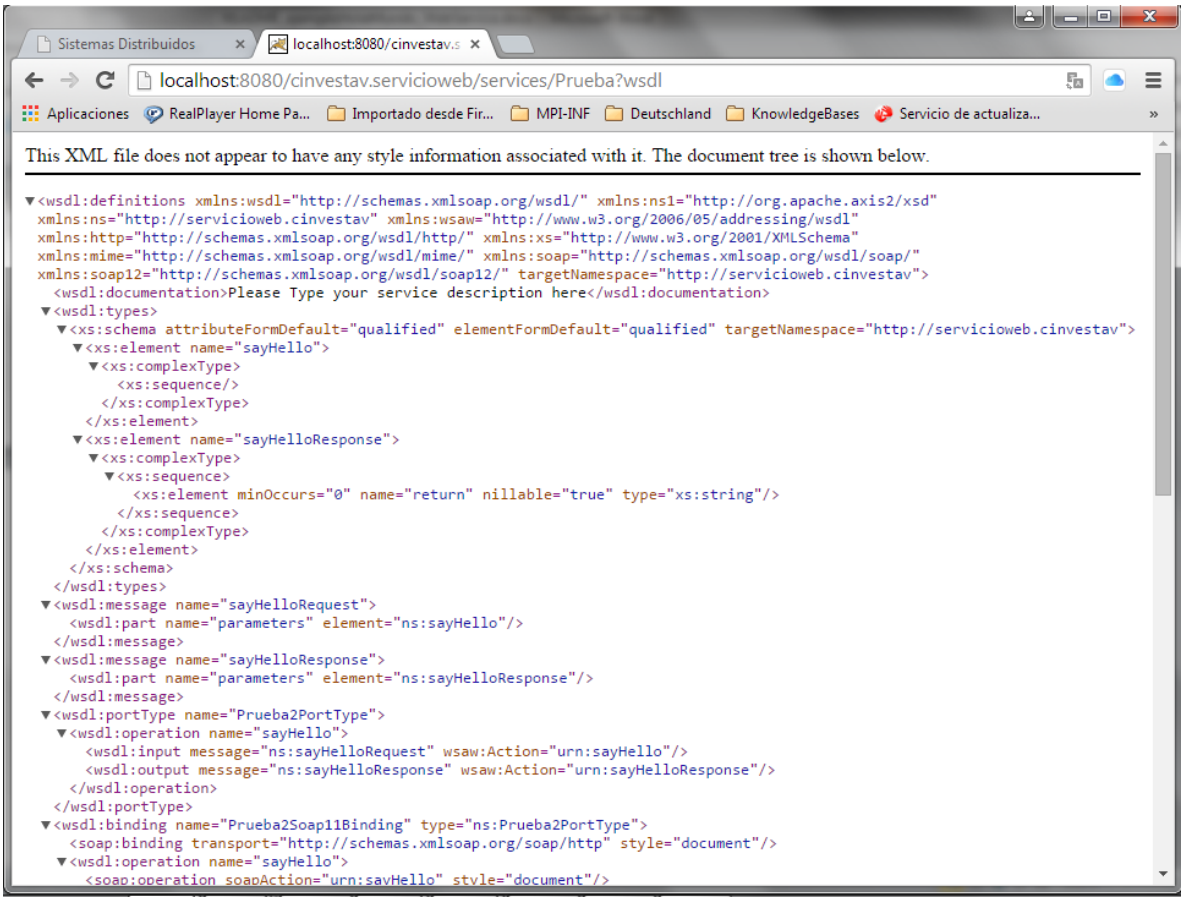
En esta dirección nos aparecerá la página de inicio de Axis2 con un enlace que pone Services. Al dar click sobre él, aparecerá el listado de todos los servicios web que hasta el momento hayan sido desplegados, por lo que debería aparecer



Al dar click en Services veremos nuestro servicio web.

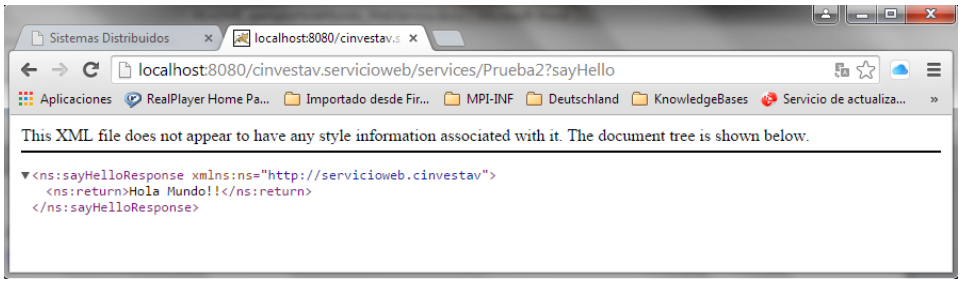


Si damos click en Prueba nos mostrará el contenido del WSDL (descripción del servicio). Podemos ver que este contenido se encuentra en la url:
http://localhost:8080/cinvestav.servicioweb/services/Prueba?wsdl



Para invocar la operación **sayHello** desde el navegador podemos emitir la siguiente url: http://localhost:8080/cinvestav.servicioweb/services/Prueba?sayHello

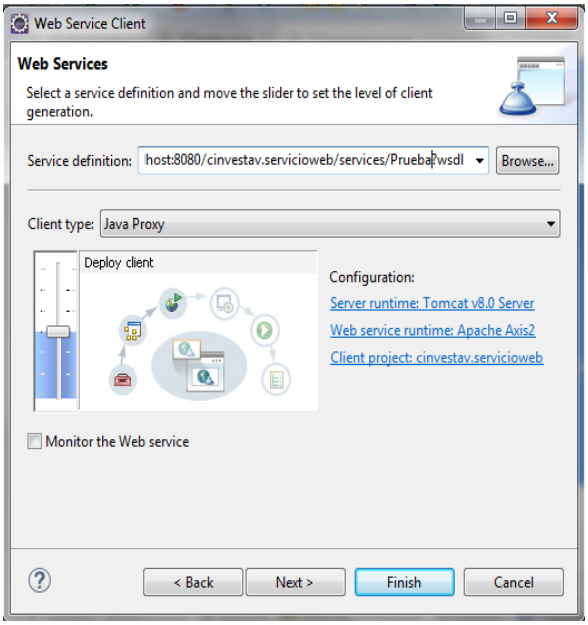
El navegador mostrará la respuesta en XML.

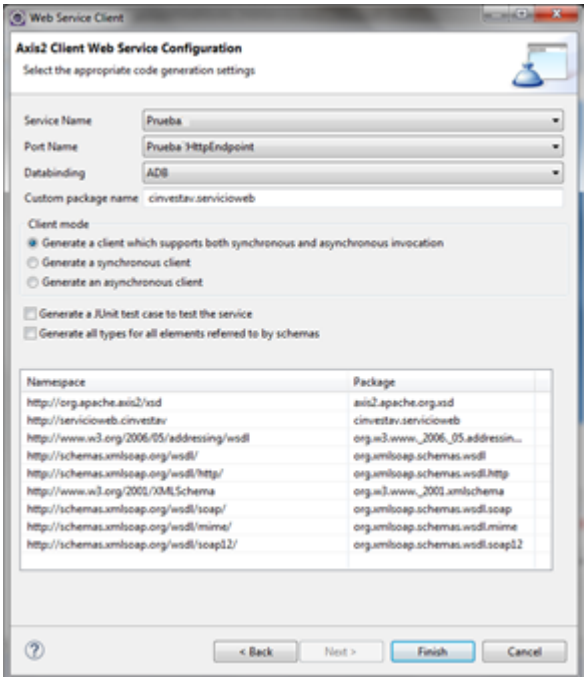


El siguiente paso será la generación de los Stubs a partir de la clase que hemos desarrollado como servicio web. Estos Stubs permitirán a los clientes invocar las operaciones definidas en nuestro servicio web. Los archivos que se generan contienen todos los elementos (clases) necesarios para invocar dichas operaciones desde una aplicación cliente; hace las veces de proxy.

Para generar los Stubs: Dar botón derecho sobre la clase Prueba y:
New -> Other- > Web Service Client.

En "Service definition" ponemos la url donde se ubica el WSDL del servicio:
http://localhost:8080/cinvestav.servicioweb/services/Prueba?wsdl





Podemos ver que el cliente puede trabajar en varios modos (Client Mode): synchronous, asynchronous o ambos. Elegimos la opción synchronous para este ejemplo, lo que quiere decir que el cliente deberá esperar por la respuesta una vez que invoque la operación. A manera de ejercicio también podríamos hacer una prueba donde el cliente trabaje en modo asynchronous o en ambos.

Una vez generado estos archivos Stub, procedemos a programar nuestra clase cliente:

```java
package cinvestav.servicioweb;

import java.rmi.RemoteException;
import org.apache.axis2.AxisFault;

import cinvestav.servicioweb.PruebaStub.SayHello;
import cinvestav.servicioweb.PruebaStub.SayHelloResponse;

public class Cliente {
	public static void main(String[] args) {
		try {
		//Se crea un objeto del proxy(Stub) hacia el servicio web
		PruebaStub holaMundo = new PruebaStub();

		//Creamos un objeto que representa la operación a invocar.
		SayHello diHola = new SayHello();

		//Se invoca la operación y se obtiene la respuesta.
		SayHelloResponse respuesta = holaMundo.sayHello(diHola);

		//Se muestra la respuesta.
		System.out.println(respuesta.get_return());

		} catch (AxisFault e) { e.printStackTrace(); }
		  catch (RemoteException e) { e.printStackTrace(); }
	}
}
```

Al ejecutar esta clase podemos ver la respuesta de la operación sayHello en pantalla. Es importante notar que para este ejemplo tuvimos que incluir:

```java
import cinvestav.servicioweb.PruebaStub.SayHello;
import cinvestav.servicioweb.PruebaStub.SayHelloResponse;
```

Paquetes que nos sirven para crear el objeto que representará a la operación a ejecutar y que recibirá la respuesta.

También los paquetes:

```java
import java.rmi.RemoteException;
import org.apache.axis2.AxisFault;
```

que se utilizan para gestionar (try-catch) los posibles errores que pudieran surgir en la conexión.