Desarrollo de Web Services Java con Rational Application Developer y WebSphere Application Server 6.0

Laboratorios

Laboratorio 1 - Creación de un servicio web a partir de un Java bean

Laboratorio 2 - Publicación de un servicio web a un directorio UDDI

Laboratorio 3 - Creación de un cliente de un servicio web partiendo de un archivo WSDL

Laboratorio 4 - Creación de un cliente web utilizando Java Server Faces (JSF)

Laboratorio 1
Creación de un servicio web a partir de un Java bean

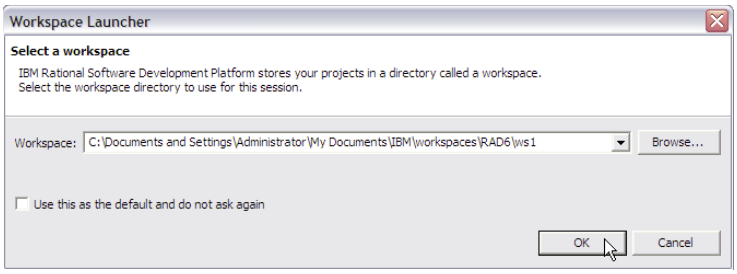
Objetivos
Uno de los principales objetivos de los impulsores de los WebServices ha sido diseñar unas especificaciones que permitieran crear rápidamente nuevos servicios a partir de componentes ya existentes y así simplificar al máximo la creación de nuevos servicios Web. Con el producto Rational Developer Studio que usaremos en los laboratorios es posible crear de manera muy sencilla servicios web a partir de componentes Java estándar tales como Java beans o Enterprise Java Beans (EJBs) o incluso stored procedures de una base de datos DB2. Con productos adicionales es posible exponer como servicios web aplicaciones de mainframe o transacciones de un ERP. Para nuestro primer laboratorio atacaremos el caso más sencillo, el desarrollo de un servicio web a partir de un Java bean.

Prerrequisitos
Rational Application Developer debe estar instalado en su equipo.

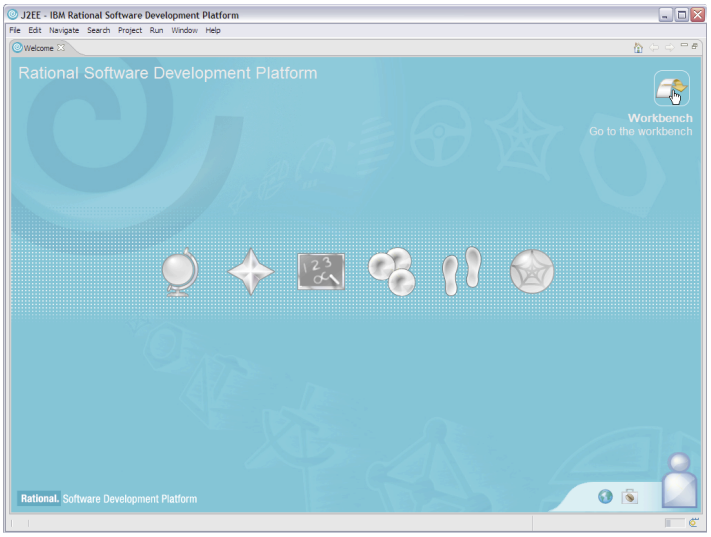
Instrucciones

Inicie el entorno de desarrollo pulsando el siguiente icono en la barra de tareas  .

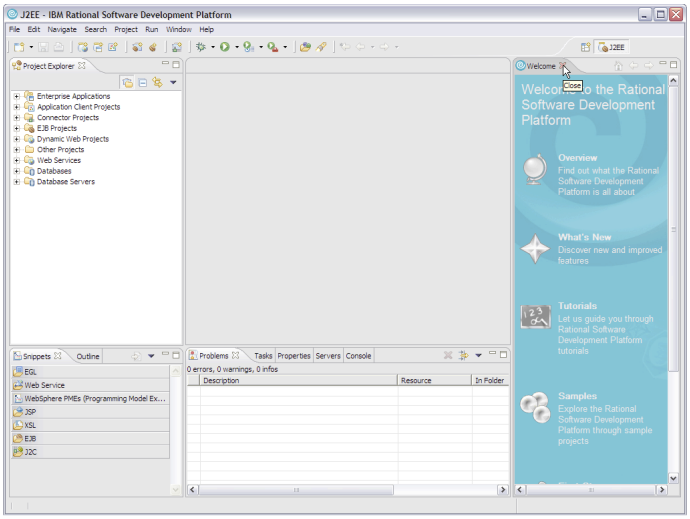
A continuación aparece una pantalla en la que se solicita la ubicación de un directorio en el que se guardarán todos los archivos del proyecto (workspace). Elija un directorio que no exista en su máquina para empezar desde cero.



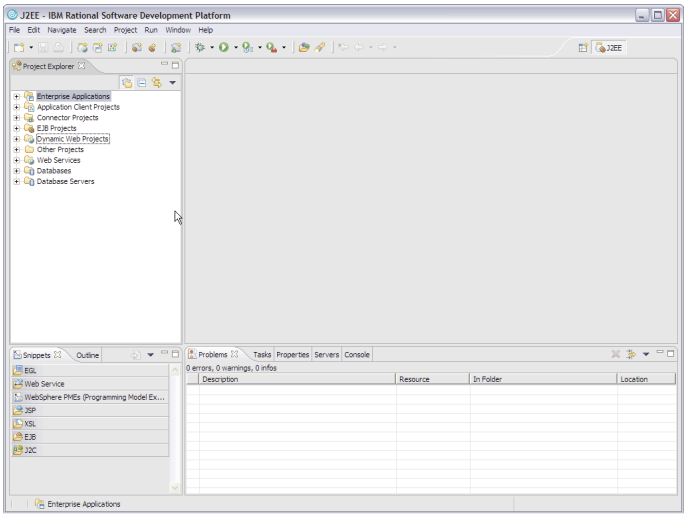
Una vez completado este paso, el entorno de desarrollo arranca y aparece la siguiente pantalla

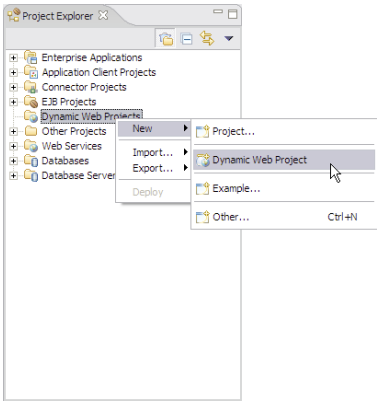


Pulse el ícono en forma de flecha situado en el extremo superior derecho de la pantalla para salir de la pantalla de bienvenida.



Cierre la pestaña “Welcome” como se muestra a continuación para poder empezar a trabajar sin que nos estorbe.

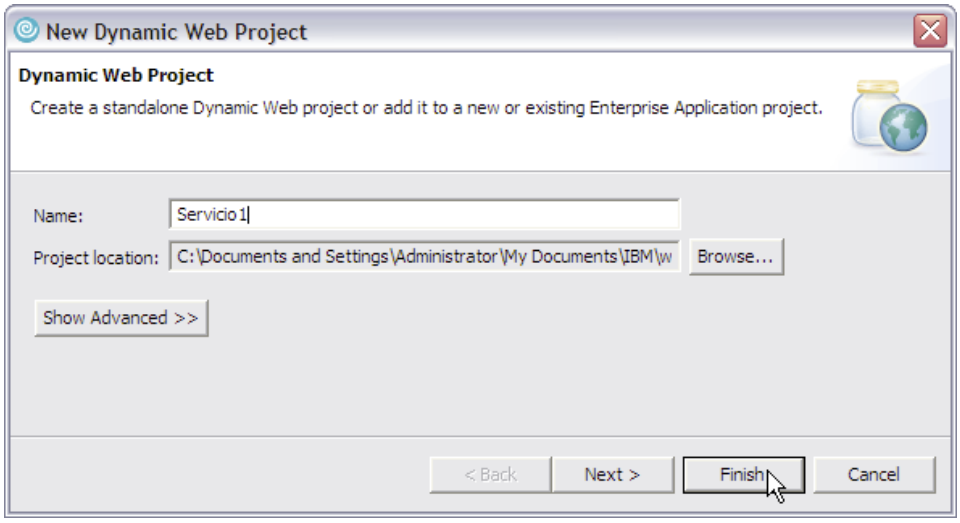




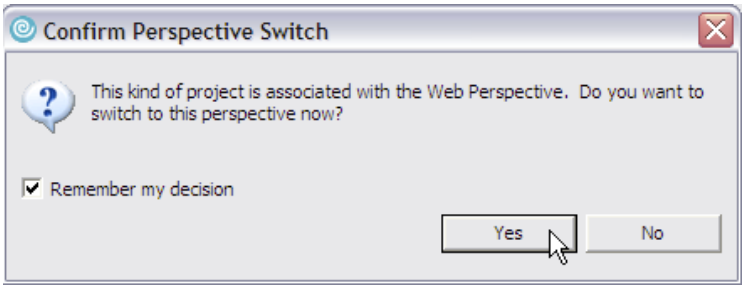
Los Java Beans son componentes sencillos que cuando hablamos de Enterprise Applications corren dentro del Web Container, a diferencia de los Enterprise Java Beans que son más complejos y que corren en su propio contenedor (EJB container).

Por lo tanto, para poder crear un Java Bean, primero debemos crear una aplicación de web dinámica. Para ello haga click con el botón derecho del mouse apretado (right-click) sobre el directorio Dynamic Web Projects dentro de la pestaña Project Explorer como se muestra en la ilustración.

Llame “Servicio1” al nuevo proyecto Web y pulse el botón “Finish” para crear el proyecto.



Como actualmente no tiene abierta la perspectiva de Web, el entorno de desarrollo le pregunta si desea abrirla. Conteste que si y marque la casilla para que el entorno de desarrollo no le vuelva a hacer la misma pregunta en el futuro.

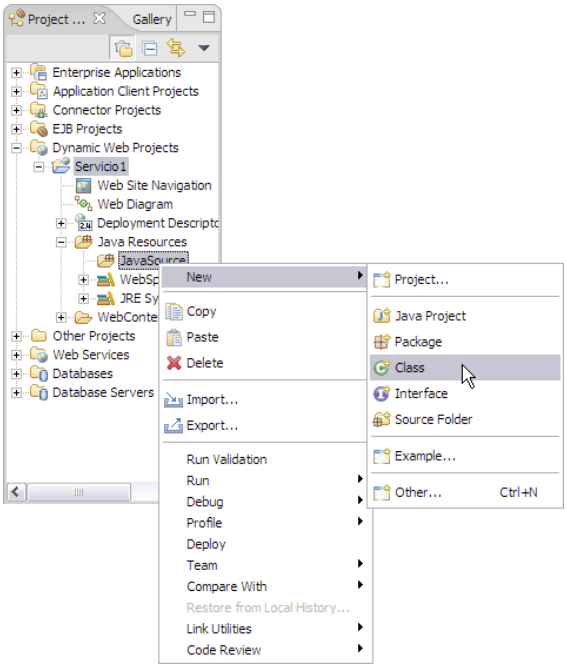


Ya estamos listos para empezar a desarrollar el Java bean que servirá de base a nuestro servicio web. Vamos a desarrollar un servicio que provea el valor en bolsa de distintas acciones. Como no tenemos acceso a datos reales, vamos a simularlos usando un valor de referencia fijo, aplicándole una variación aleatoria de +/- 5%.

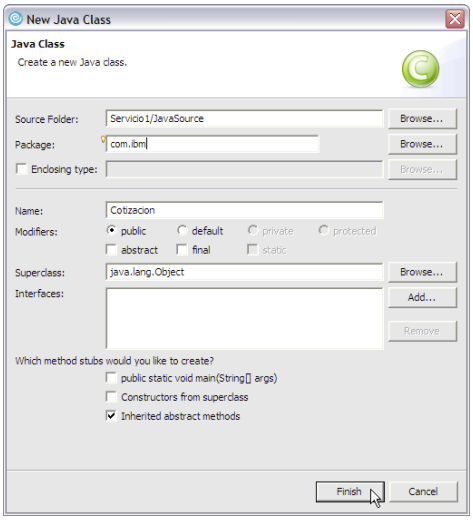
Nuestro servicio web va a recibir un parámetro de entrada, una cadena de caracteres que representa el nombre de la empresa (por ejemplo AAPL para Apple Computer). El web service nos va a regresar un solo resultado, un objeto de tipo Cotizacion que tiene dos atributos, nombre de tipo String (cuyoo valor es el del parámetro de entrada) y valor de tipo float que va acontener el valor ficticio de una acción de la empresa seleccionada. Si la empresa no existe, el valor valdrá -1.

En principio siempre es una buena práctica desarrollar web services que regresan como parte del resultado uno, varios o incluso todos los valores de entrada. Esto permite que si los servicios son invocados dentro de un proceso se simplifique la tarea de correlacionar resultados con invocaciones.

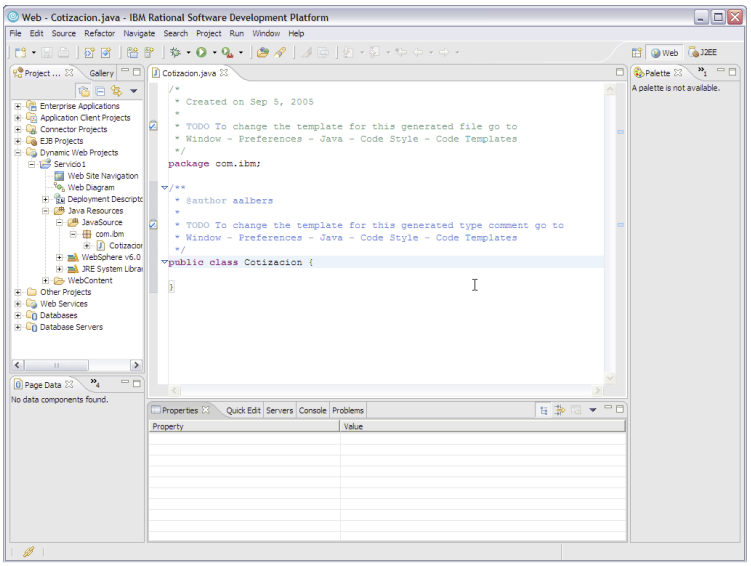
Para implementar nuestro servicio de cotizaciones debemos crear dos Java beans. El primero es el que representa el objeto Cotizacion que el servicio usa para regresar el resultado. Para crearlo vamos a usar el wizard qe simplifica la creación de clases. Expanda el proyecto web (“Servicio1”) en la pestaña de “Project Explorer” haciendo click sobre el símbolo + situado a la derecha del nombre del proyecto. Ahora haga right-click sobre la carpeta JavaSource que es la que se usa para almacenar las clases Java del proyecto Web. Seleccione del menú contextual la opción New >> Class como se muestra a continuación.



Esto tiene como resultado lanzar el wizard. Vamos a indicar que la nueva clase va a pertenecer al paquete “com.ibm” y que se va a llamar “Cotización”. Deje sin tocar todos los demás campos de la ventana y pulse “Finish” para continuar.



Al terminar el proceso, el editor le debe mostrar la nueva clase “Cotizacion” tal y como se muestra a continuación.



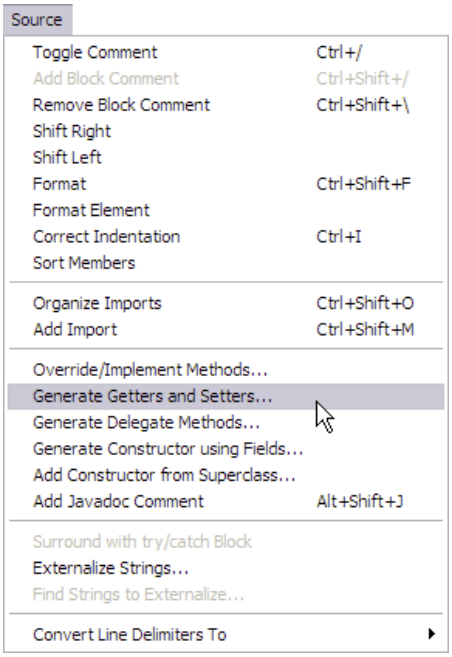
Dentro del cuerpo de la clase vamos a añadir las dos variables de instancia y el constructor.

```
String nombre;
float valor;

public Cotizacion(){
}

public Cotizacion(String nombre, float valor){
```

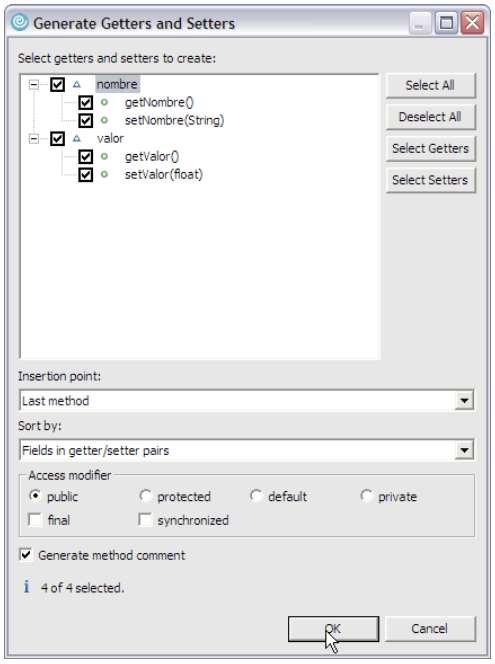
```
        this.nombre = nombre;
        this.valor = valor;
}
```

El constructor vacío es necesario para que la herramienta pueda construir correctamente el XML Schema.

Para manipular las variables de instancia nos falta crear los getters/setters. Vamos a encargar a RAD que lo haga automáticamente. Situe el cursor justo antes de la llave (el símbolo "}") que cierra la definición de la clase. Bajo el menú "Source" seleccione la opción "Generate Getters and Setters...".



Ahora seleccione todas las opciones para que se generen tanto los getters como los setters para ambos atributos.



Con esta operación hemos terminado el trabajo sobre la clase "Cotizacion". Ahora vamos a crear la clase "Servicio", la cual va a ser la base del servicio que estamos desarrollando.

Utilizando el mismo método que en el caso anterior, cree la clase "Servicio" dentro del paquete "com.ibm".

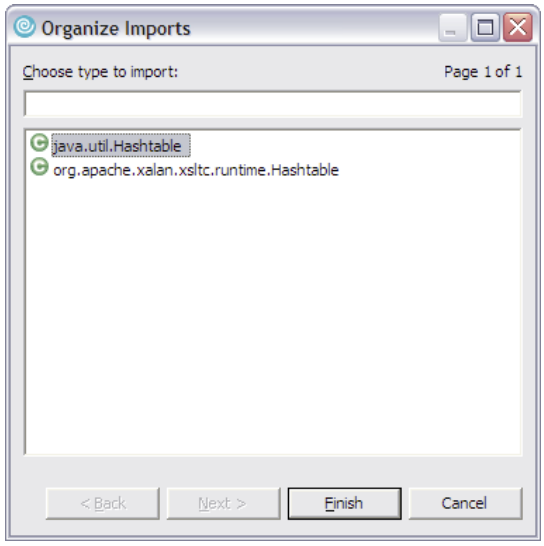
Esta clase solo va a tener un método llamado "obtenerCotizacion" y no va a usar variables de instancia para almacenar valores intermedios. Es importante que recuerden que cuando convirtamos la clase "Servicio" en un servicio web esta va a ser invocada desde un servlet. Por lo tanto, todas las reglas que aplicamos en un servlet para evitar problemas de concurrencia, también aplican aqui si no desean obtener resultados impredecibles.

Agregue el siguiente código a la clase recién creada

```
Hashtable   valores = new Hashtable();
Random      generador = new Random();
```

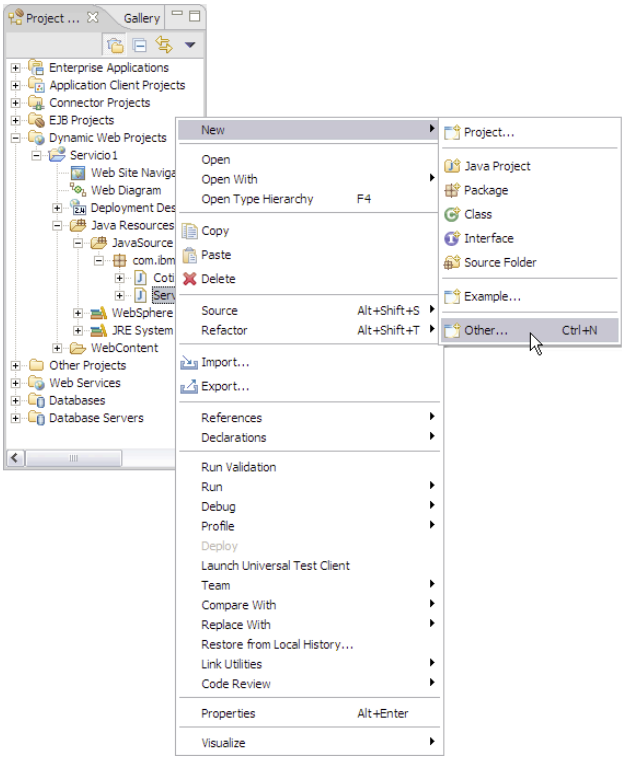
```
public Cotizacion obteberCotizacion(String nombre){
float       resuktado = -1;
Object      tempObject;

      if (valores.isEmpty()){
            valores.put("IBM",new Float(83));
            valores.put("AAPL",new Float(43));
            valores.put("ORCL",new Float(13));
            valores.put("HPQ",new Float(27));
            valores.put("MSFT",new Float(26));
      }

      tempObject = valores.get(nombre.toUpperCase().trim());
      if (tempObject != null){
            resultado = ((Float)tempObject).floatValue();
            resultado *= (0.975 + generador.nextFloat()/20);
      }

      return new Cotizacion(nombre,resultado);
}
```

El código debe marcar errores porque faltan importar algunos paquetes. Deje que RAD se encargue de eso. Pulse Ctrl-Shift-O (alternativamente puede seleccionar la opción "Organize Imports" del menú "Source"). En algunos casos puede haber duda sobre el paquete que se necesita importar. Seleccione el correcto ("java.util.Hashtable"). Los errores deben haber desaparecido. Guarde sus cambios pulsando Ctrl-S.

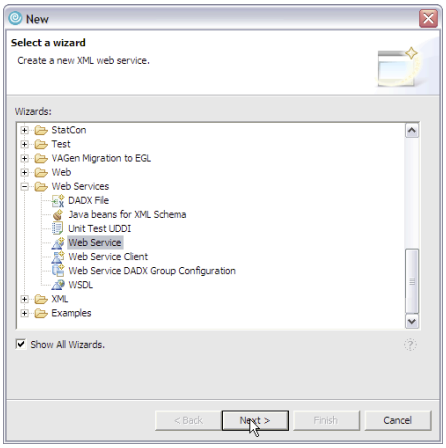


Ya estamos listos para convertir el método "obtenerCotizacion" de la clase "Servicio" en un web service. Eso es importante ya que el servicio web no es la clase sino el método. Por lo tanto, de una clase con varios métodos se pueden derivar múltiples servicios web. Normalmente, cuando ese es el caso, se genera sin embargo un solo archivo WSDL que describe todos los servicios.

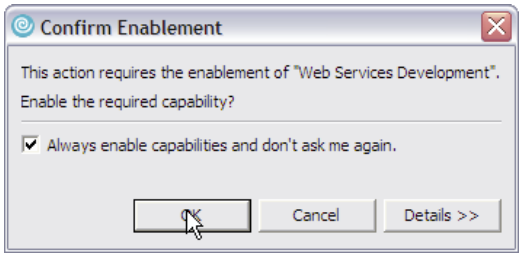
Para crear el web service vamos a usar un wizard (asistente) incluído en el Rational Application Developer (RAD). Haga right-click sobre la clase “Servicio”. Del menú contextual seleccione la opción New>>Other tal y como se muestra en la siguiente ilustración.



Seleccione la opción “Show all wizards” y posteriormente elija “Web Service” dentro de la carpeta “Web Services” y pulse “Next”.



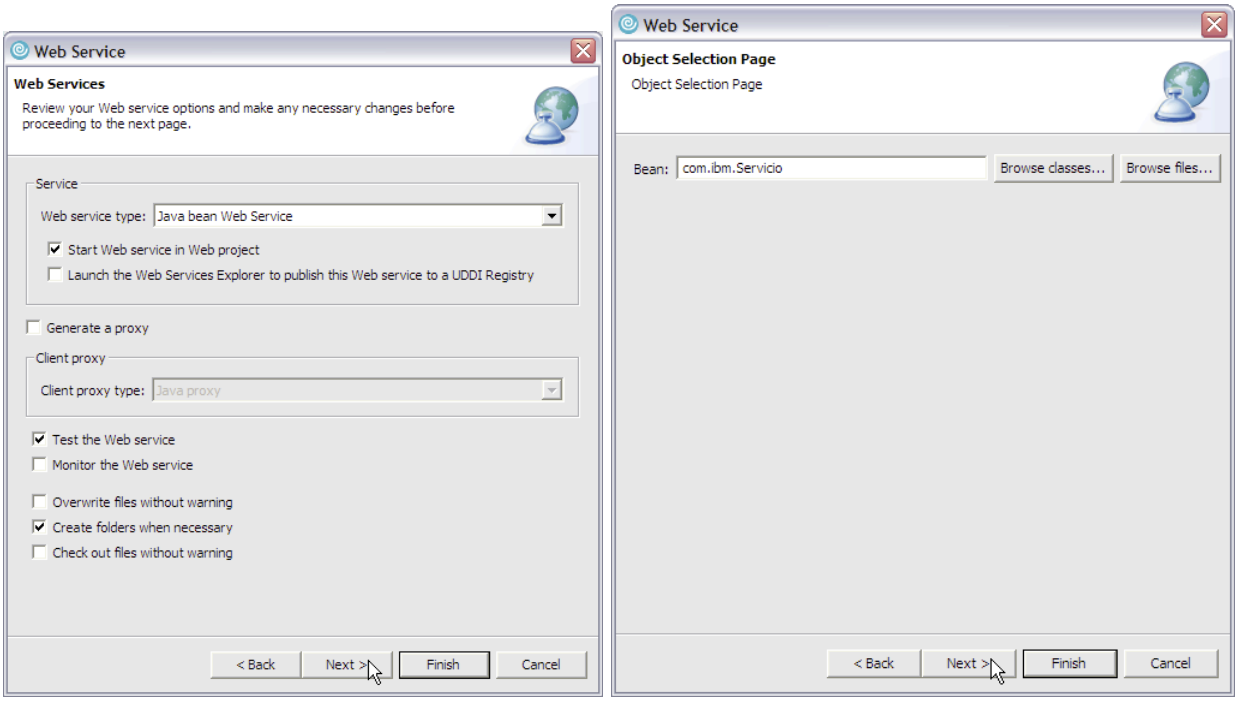
Como es la primera vez que se crea un servicio web dentro de este workspace, RAD solicita si desea activar esta funcionalidad ya que esta consume memoria y está deshabilitada por defecto. Seleccione la opción “Always enable capabilities and don’t ask me again/” y pulse “OK”.



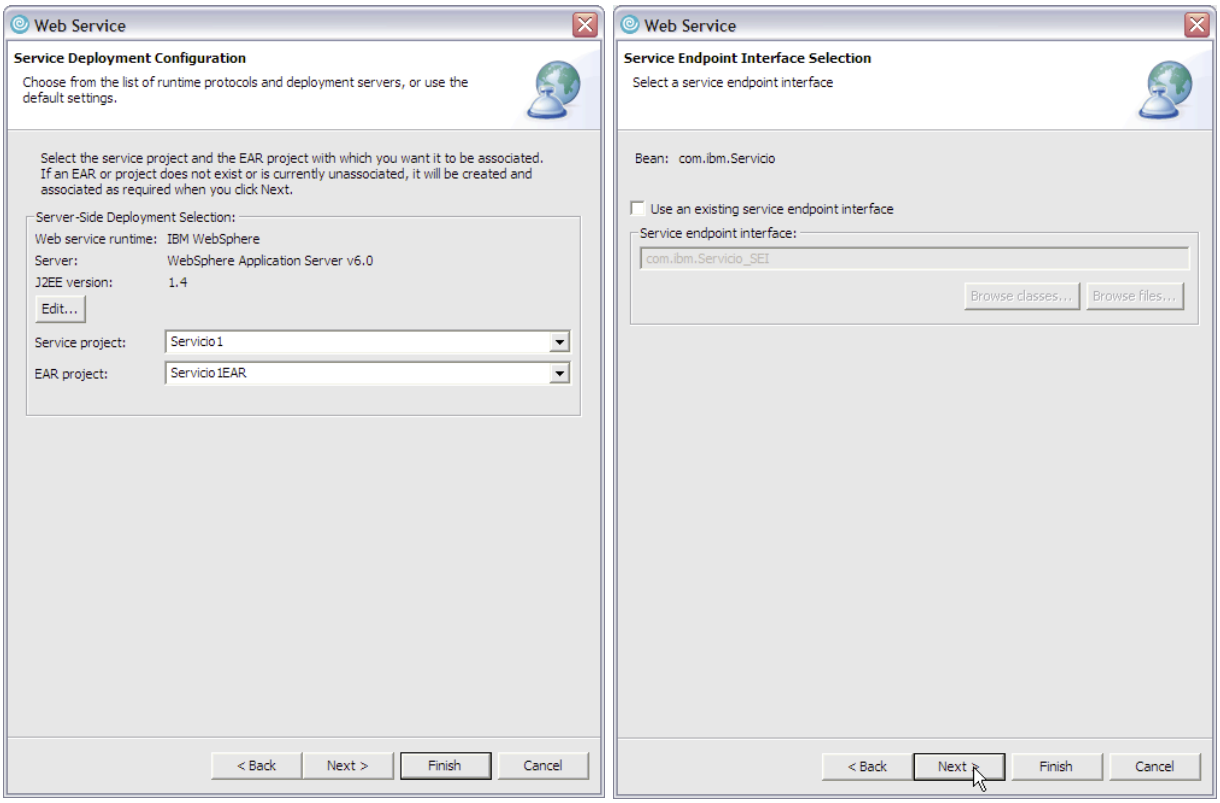
Las siguientes páginas del asistente son necesarias para recabar la información necesaria a la generación del servicio web.

En la primera página ya está marcada la opción que indica que queremos crear un servicio web a paratir de un Java bean. Por lo tanto, lo único que tenemos que hacer es marcar la opción “Test the Web Service” para que al final sea posible probar el servicio sin necesidad de escribir un cliente. Pulse “Next>>”.

En la segunda página no hace falta cambiar nada porque ya está seleccionada la clase que contiene el servicio (eso se debe a que hicimos right-click sobre el nombre de la clase al arrancar el asistente. Pulse “Next>>”.

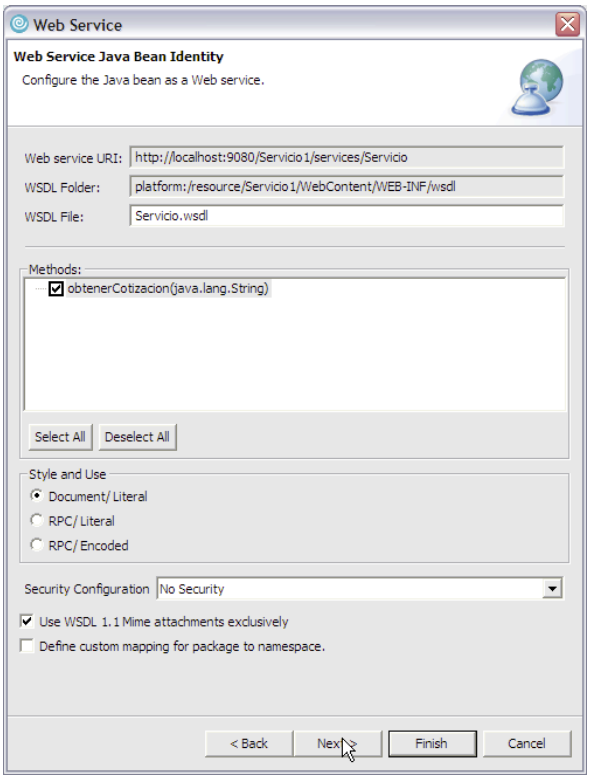


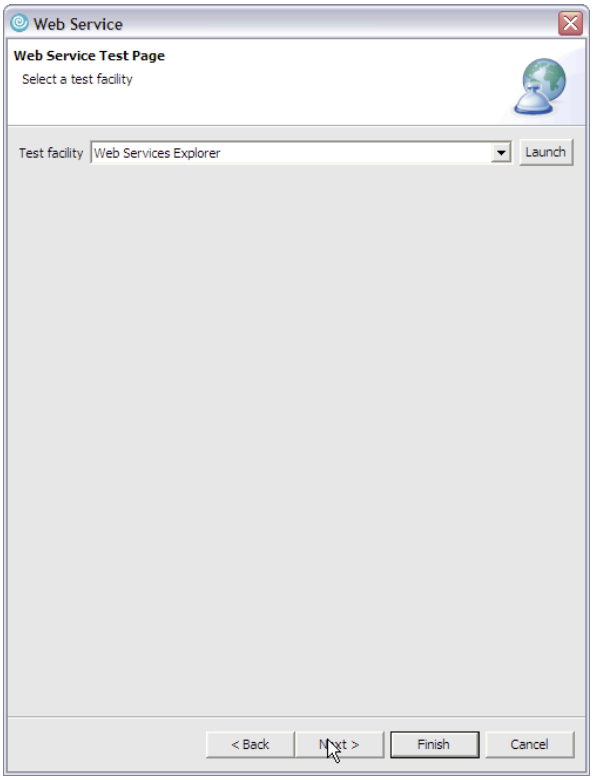
La tercera página tampoco requiere cambios. Simplemente pulse “Next >>”. Lo mismo aplica a la página cuatro. Pulse de nuevo el botón “Next >>”.



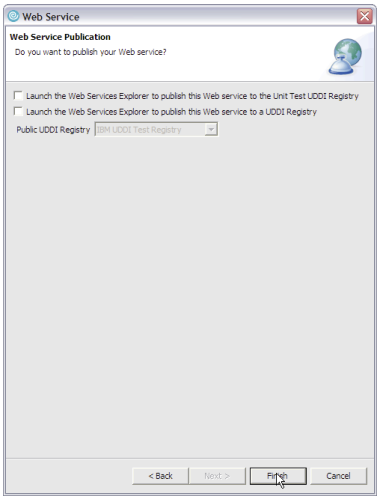
En la siguiente página se presentan los distintos métodos del Java Bean seleccionado que van a ser convertidas a servicios web. En este caso, como el bean “Servicio” solo tiene un método (“obtenerCotizacion”) y este ya está seleccionado no tenemos necesidad de cambiar nada. Pulse de nuevo el botón “Next >>”.

La penúltima pantalla tampoco requiere de ningún cambio ya que vamos a usar el “Web Services Explorer” para hacer las pruebas. De nuevo, pulse el botón “Next >>”.





Finalmente llegamos a la última pantalla del asistente. En este laboratorio no vamos a publicar automáticamente el servicio en el directorio UDDI. Eso es algo que haremos manualmente en el siguiente ejercicio. Por lo tanto, solo pulse el botón “Finish” para terminar.



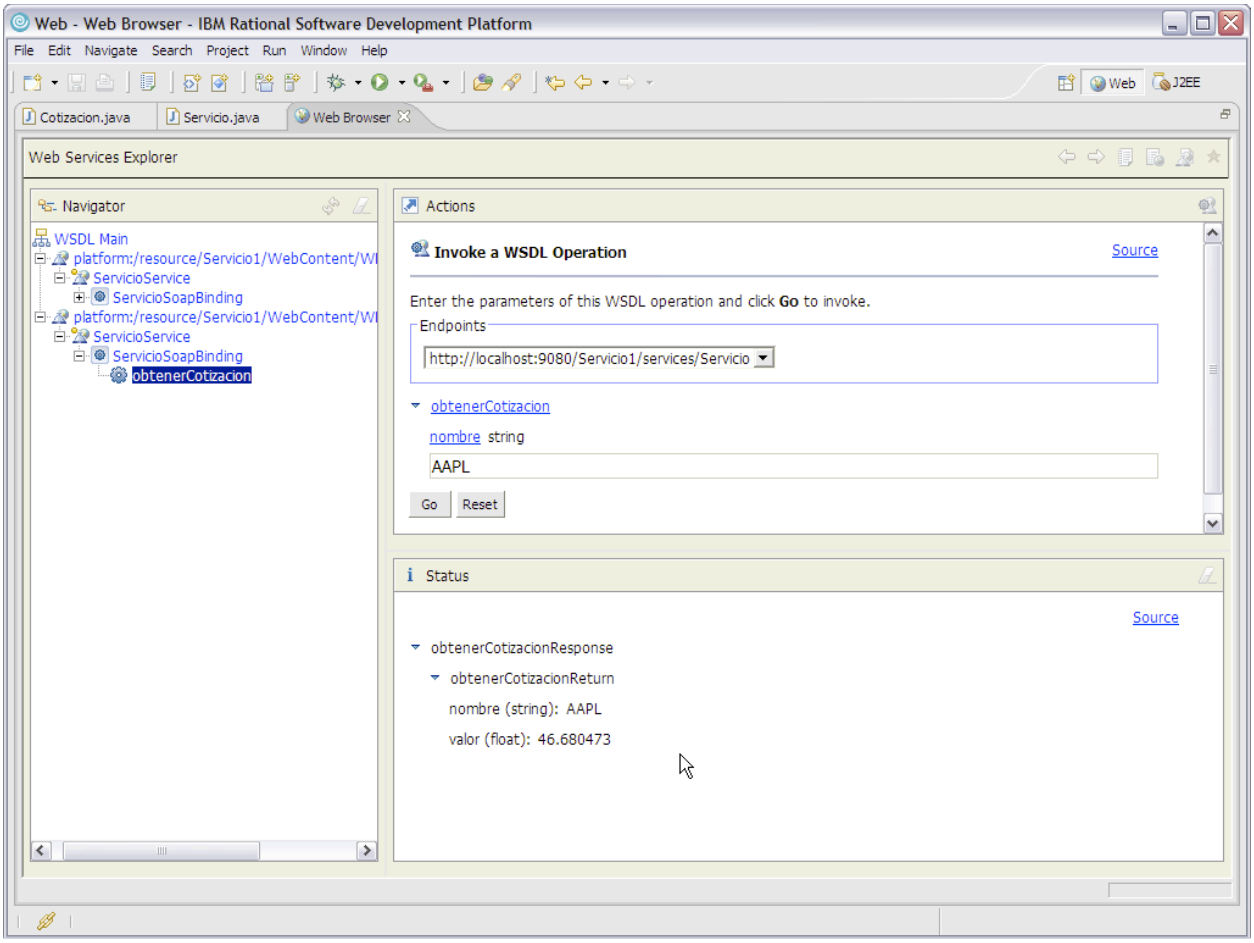
En ese momento inicia la generación automática de las clases necesarias para la ejecución del servicio web asi que el fundamental archivo WSDL que describe el servicio.

Las clases generadas se encuentran dentro del paquete “com.ibm”. El archivo WSDL está situado en el directorio WebContent/wsdl. Más adelante revisaremos con más detalle estos archivos.

El asistente arranca automáticamente el servidor de aplicaciones para poder probar el servicio. Cuando esta operación se termina satisfactoriamente, se abre una nueva pestaña que contiene un navegador web para acceder a la aplicación “Web Services Explorer”.

Esta aplicación es la que se muestra a continuación. Para pode usarla con mayor facilidad es recomendable que hagan doble click sobre el título de la pestaña (“Web Services Explorer”). Esto tendrá por efecto

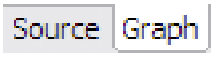
maximizar la ventana del navegador. Pruebe el servicio usando distintos valores (“AAPL”, “IBM”, etc.). Verifique que funciona correctamente.



Minimice la ventana haciendo doble-click sobre la pestaña “Web Browser”. Vamos a examinar los archivos creados por el asistente. Empecemos por el más importante, el archivo WSDL.

Como la clase que creamos para el servicio web se llama Servicio y está situada en el paquete “com.ibm”, el archivo WSDL generado se encuentra en Dynamic Web Projects/Servicio1/Web Content/wsdl/com/ibm y se llama Servicio.wsdl.

Haga doble-click sobre el nombre del archivo para abrirlo en un editor gráfico. Este editor es muy práctico para entender la estructura de un archivo WSDL ya que se trata de un archivo bastante complejo en formato XML. Si lo desdea puede alternar entre la vista gráfica y el XML haciendo click sobre las siguientes pestañas:



En realidad el formato de un archivo WSDL no es tan complejo. Lo que hay que entender es que puede estar formado de hasta cuatro partes.

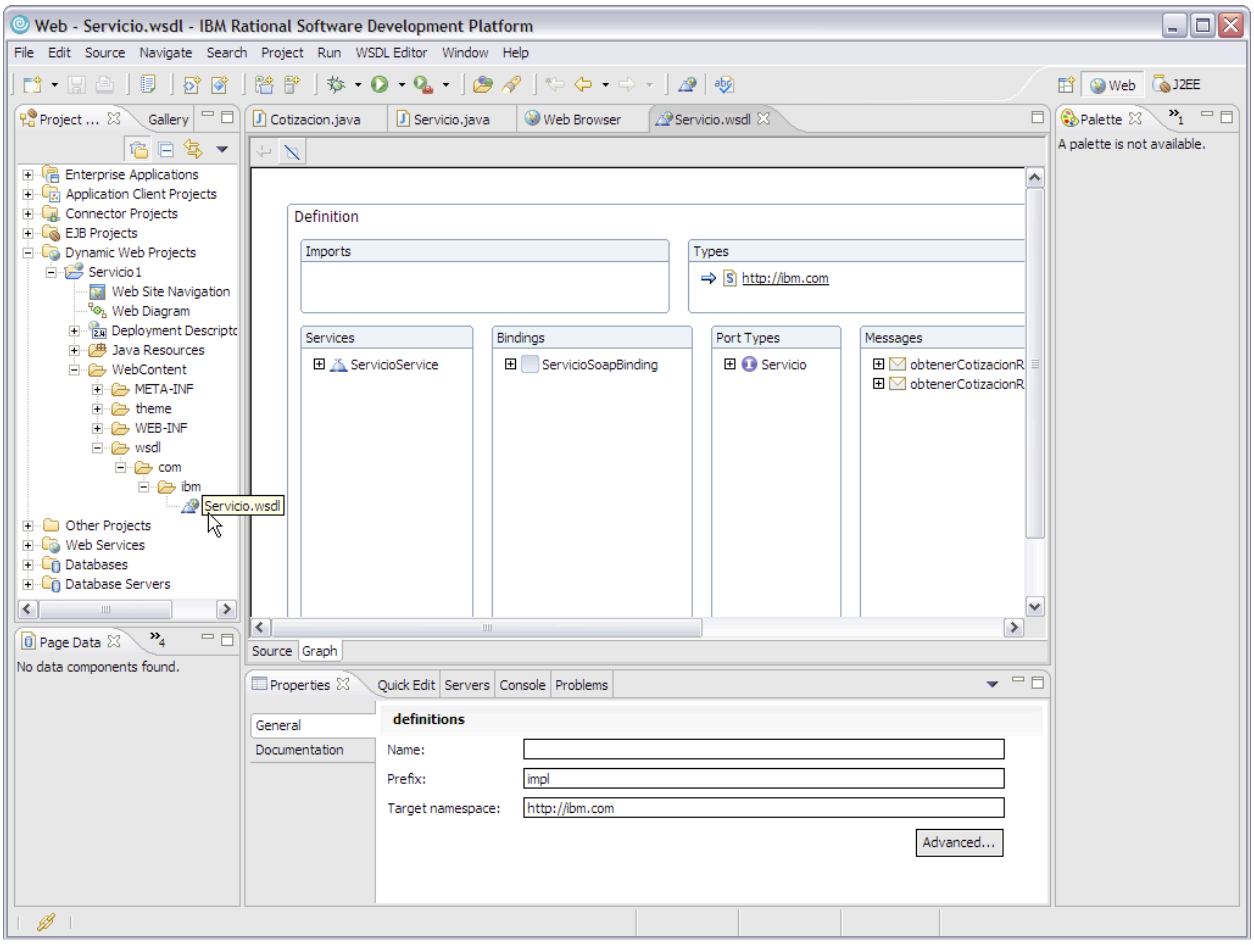
La parte de servicios contiene la definición de uno o más servicios. Un servicio no se refiere a un servicio web sino a un conjunto de servicios web que comparten determinadas características como por ejemplo

el servidor en el que se están ejecutando. Un mismo servicio puede ser implementado de varias maneras y si ese fuera el caso esto se vería reflejado en esta parte del archivo WSDL.

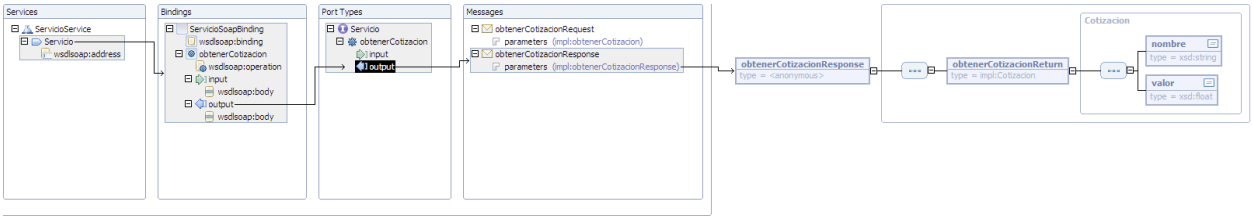
La parte de Bindings contiene información que relaciona una implementación específica con una operación (el servicio web) y sus parámetros de entrada y salida.

En el segmento de Port Types se definen las operaciones y sus parámetros. Eso permite que un Port Type pueda ser reutilizado por varios Bindings.

Finalmente, los Messages representan todos los mensajes intercambiados y están definidos utilizando el estándar XML-Schema. Lo bueno de tener separados en el archivo WSDL los mensajes de los tipos de puertos es que un mensaje puede ser el resultado de una operación pero también el argumento de entrada de otro. De esta manera, los mensajes pueden ser reutilizados.



Para entender mejor la estructura de un archivo WSDL es recomendable que maximice la ventana del editor haciendo doble-click sobre el título de la misma (“Servicio.wsdl”). Ahora explore las relaciones existentes entre las distintas partes del archivo y el contenido de las mismas. Vea por ejemplo como la operación “obtenerCotizacion” tiene un mensaje de salida de tipo “Cotizacion” (en realidad una sequencia de datos en terminología XML Schema) o como esa operación está relacionada con el servicio “ServicioService” a través del Binding ServicioSoapBinding.



Finalmente vamos a examinar los archivos Java que se generaron de manera automática en el paquete “com.ibm” dentro del directorio Java Source. En total son cuatro.

Tres de ellos, Cotizacion_Deser.java, Cotizacion_Helper.java y Cotizacion_Ser.java fueron generados para permitir el intercambio de datos de tipo Cotizacion, el cual no puede ser representado automáticamente por un tipo básico de XML-Schema. En general, por cada tipo complejo utilizado en sus servicios web se generarán este tipo de archivos.

El último, Cotizacion_SEI.java es el más importante. Este archivo es el Service Endpoint Interface que describe la implementación del servicio web.

Ninguno de estos archivos debe ser modificado manualmente.

Sin duda a estas alturas aún quedan dudas de cómo funcionan los servicios web. Por ejemplo:

¿Cómo sabe WebSphere cuántos web services han sido creados?

Todo esa información es administrada automáticamente por WebSphere y Rational Application Developer. Si observa el contenido de la carpeteta WebContent/WEB-INF verá un archivo llamado webservices.xml. Este archivo asi como otros que se encuentran dentro del mismo directorio son los responsables de la “magia”. En general no es necesario que modifique esos archivos, a menos de que desee por ejemplo borrar un servicio.

Laboratorio 2
Publicación de un servicio web a un directorio UDDI

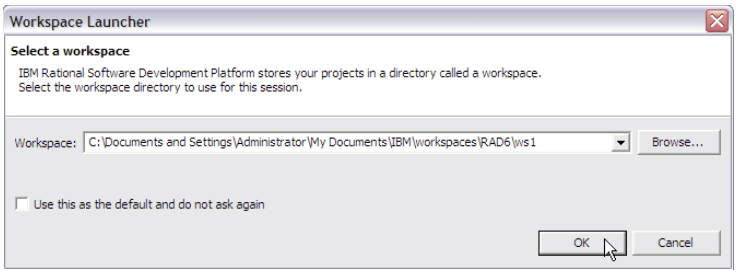
Objetivos
Ya hemos visto que crear un servicio web es algo realmente sencillo. Por lo tanto, el reto para las empresas no es la creación de los mismos sino su correcta explotación. ¿Cómo aseguramos que los programadores reutilizan servicios existentes? La respuesta es sencilla, publicandolos en un directorio público. Este directorio es parte de la especificación de Web Services y se conoce como directorio UDDI (Universal Description, Discovery and Integration). Existen dos tipos de directorios, los privados que se usan dentro de la intranet y los públicos para uso general

En este laboratorio veremos cómo explotar el directorio público que tiene IBM en Internet. El funcionamiento de ese directorio es muy similar al registro privado que incluyen tanto Rational Application Developer como WebSphere Application Server ND.

Prerrequisitos
Haber completado exitosamente el laboratorio 1.

Instrucciones
Inicie el entorno de desarrollo pulsando el siguiente icono en la barra de tareas  .

Elija el workspace creado en el laboratorio anterior.



Ahora veremos como publicar un servicio en el registro público de IBM (IBM UDDI Business Test Registry) el que es usado habitualmente para la publicación y búsqueda de servicios.

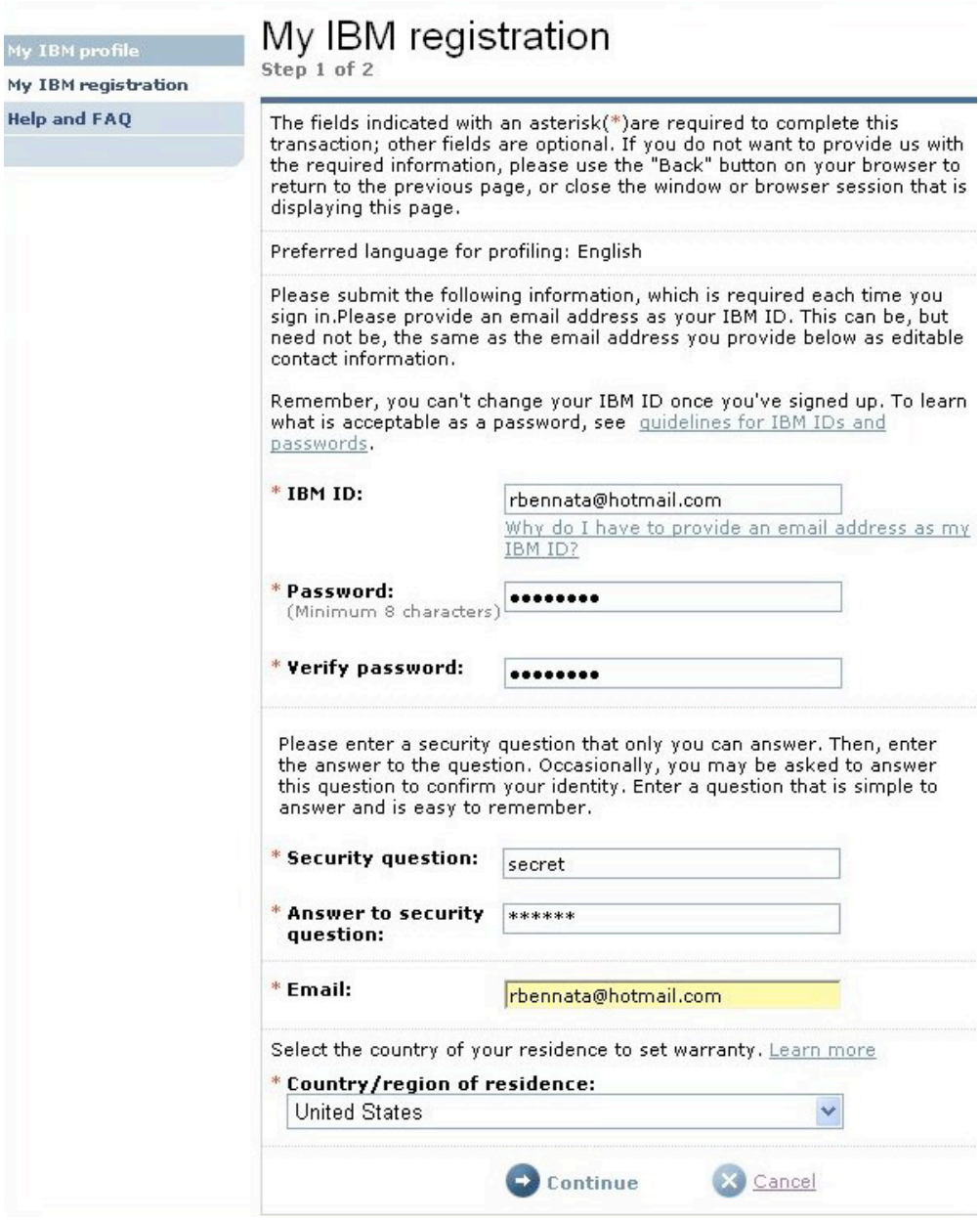
Antes de poder publicar su servicio en ese directorio deben registrarse. La clave obtenida le permitirá acceder también a otros servicios de IBM sin necesidad de volver a registrarse.

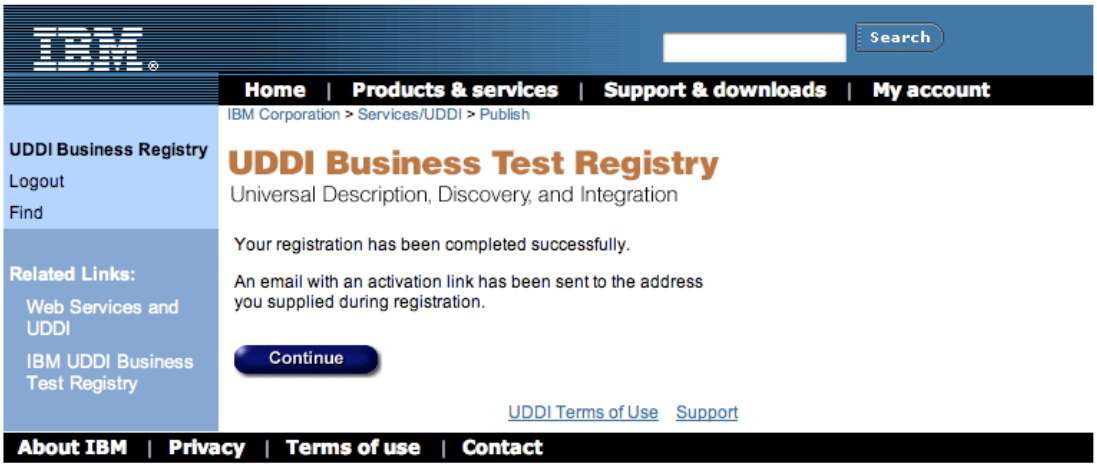
Conéctese a https://uddi.ibm.com/testregistry/registry.html.

Haga click sobre la liga “Get an IBM user ID and password”.

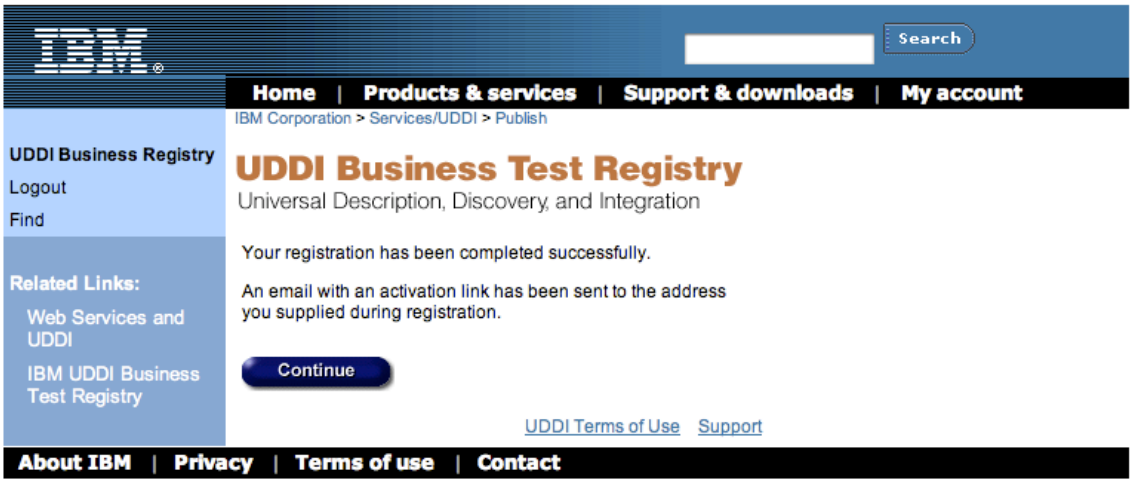
Rellene la página de registro con sus datos personales y pulse “Continue”.

A continuación aparecerá una página con los términos de uso del registro UDDI público de IBM. Pulse el botón Agree para aceptar y continuar.


My IBM profile
My IBM registration
Help and FAQ
My IBM registration
Step 1 of 2
The fields indicated with an asterisk(*)are required to complete this transaction; other fields are optional. If you do not want to provide us with the required information, please use the "Back" button on your browser to return to the previous page, or close the window or browser session that is displaying this page.
Preferred language for profiling: English
Please submit the following information, which is required each time you sign in.Please provide an email address as your IBM ID. This can be, but need not be, the same as the email address you provide below as editable contact information.
Remember, you can't change your IBM ID once you've signed up. To learn what is acceptable as a password, see guidelines for IBM IDs and passwords.
* IBM ID:
rbennata@hotmail.com
Why do I have to provide an email address as my IBM ID?
* Password:
(Minimum 8 characters)
* Verify password:
Please enter a security question that only you can answer. Then, enter the answer to the question. Occasionally, you may be asked to answer this question to confirm your identity. Enter a question that is simple to answer and is easy to remember.
* Security question:
secret
* Answer to security question:
* Email:
rbennata@hotmail.com
Select the country of your residence to set warranty. Learn more
* Country/region of residence:
United States
Continue
Cancel

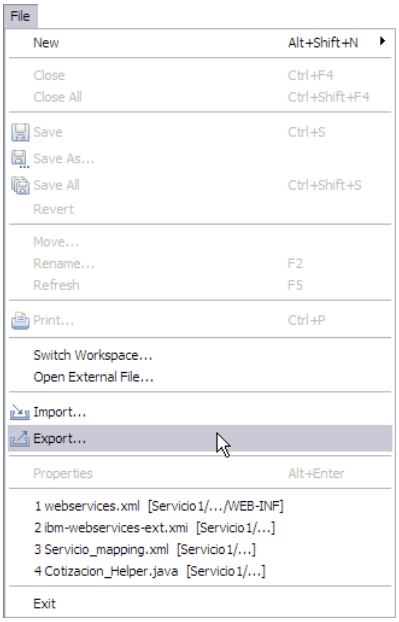

IBM
Search
Home | Products & services | Support & downloads | My account
IBM Corporation > Services/UDDI > Publish
UDDI Business Registry
Logout
Find
Related Links:
Web Services and UDDI
IBM UDDI Business Test Registry
UDDI Business Test Registry
Universal Description, Discovery, and Integration
Your registration has been completed successfully.
An email with an activation link has been sent to the address you supplied during registration.
Continue
UDDI Terms of Use Support
About IBM | Privacy | Terms of use | Contact

El sistema manda automáticamente un correo con una liga que hay que abrir para poder terminar el proceso de registro.

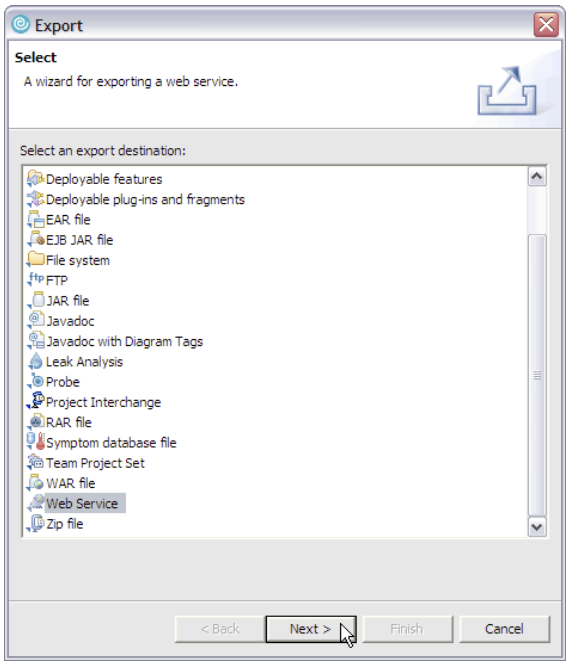


Ahora ya podemos empezar a usar el registro UDDI. Podríamos publicar el servicio usando el browser pero en lugar de eso lo haremos usando un asistente del Rational Application Developer.

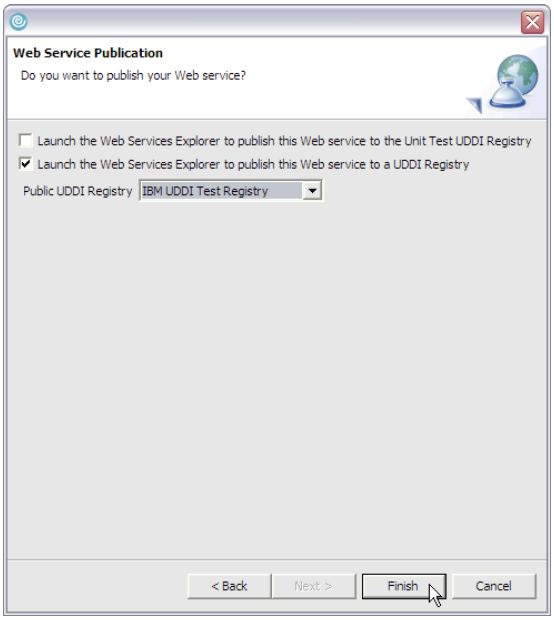
Seleccione la opción File>Export de la barra de menus.



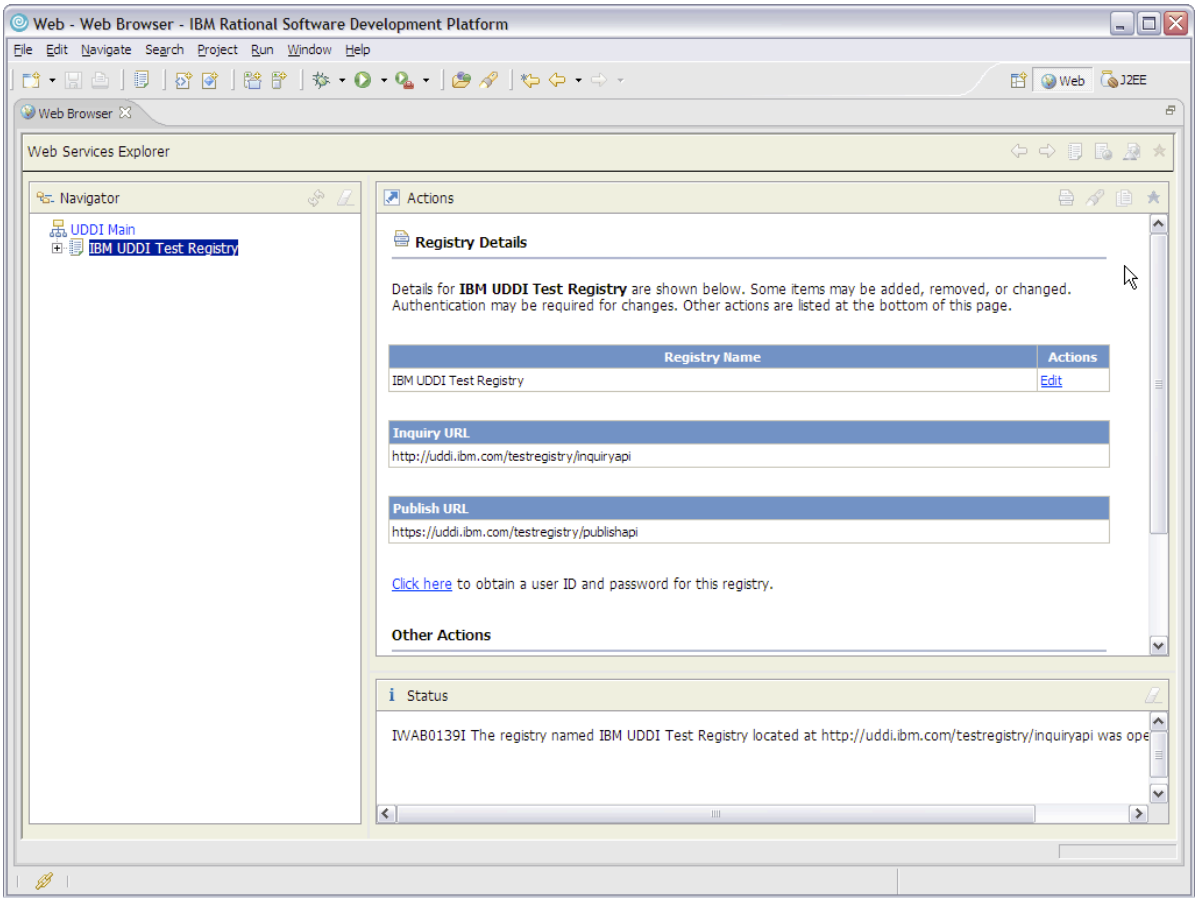
Seleccione “Web Service” de la lista de opciones que propone Rational Application Developer.



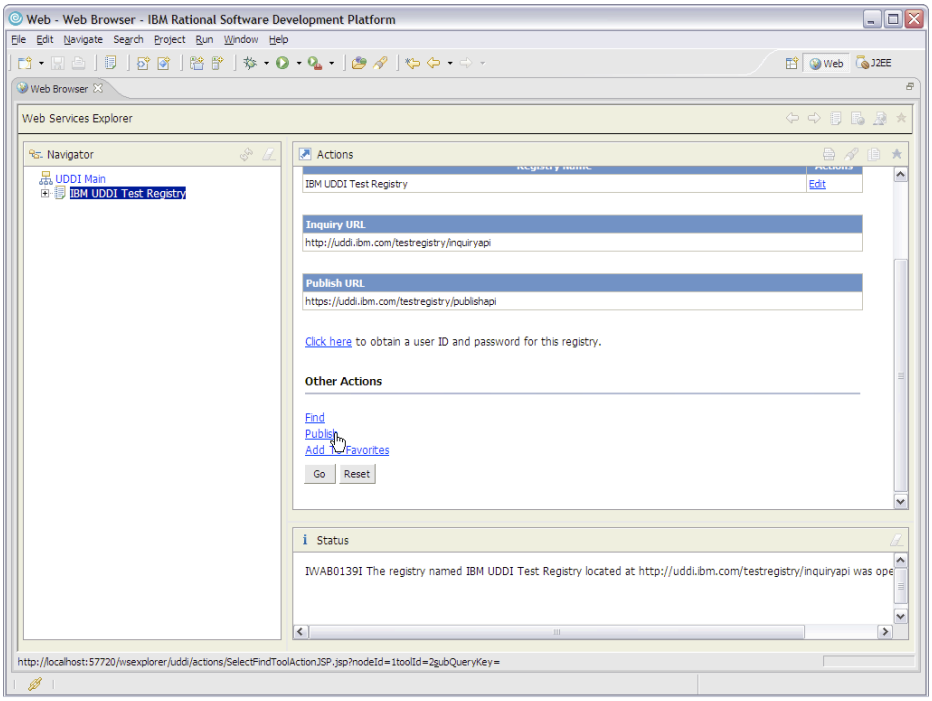
A continuación debe elegir el directorio en el que desea publicar su servicio. Puede elegir entre el directorio privado de WebSphere incluído en RAD o, como lo vamos a hacer, un directorio privado. RAD soporta varios directorios de distintos fabricantes. Nosotros vamos a usar el de IBM.



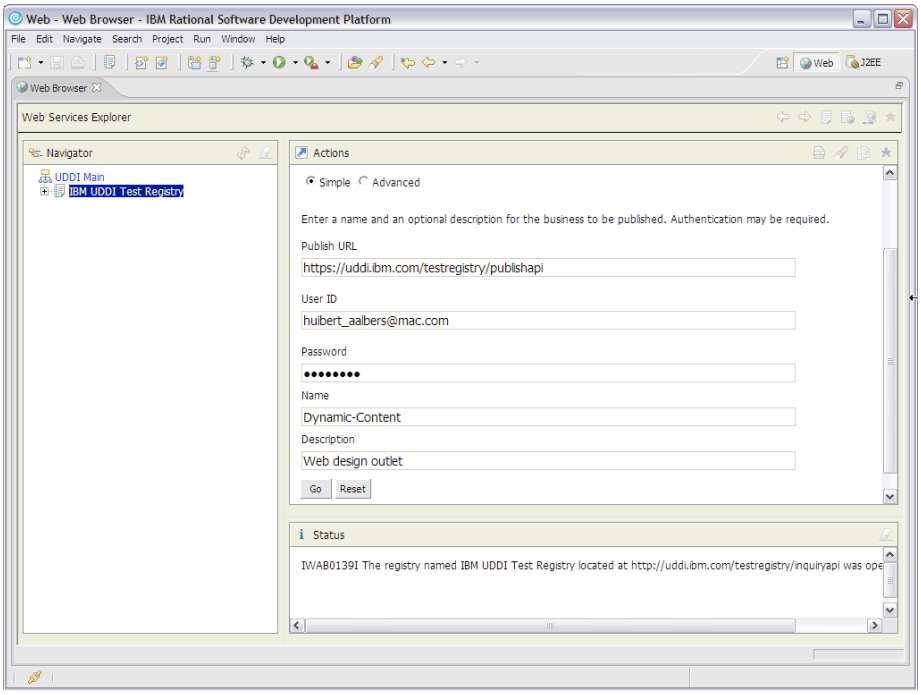
Pulse “Finish”. Esto tiene como consecuencia abrir el “Web Services Explorer”. Maximice la ventana, haciendo doble-click sobre la pestaña de la misma (“Web Browser”).



Antes de poder publicar el servicio debemos dar de alta a la empresa que lo ofrece. Para hacerlo, en la parte de abajo de la ventana “Actions” y haga click sobre la liga “Publish”.

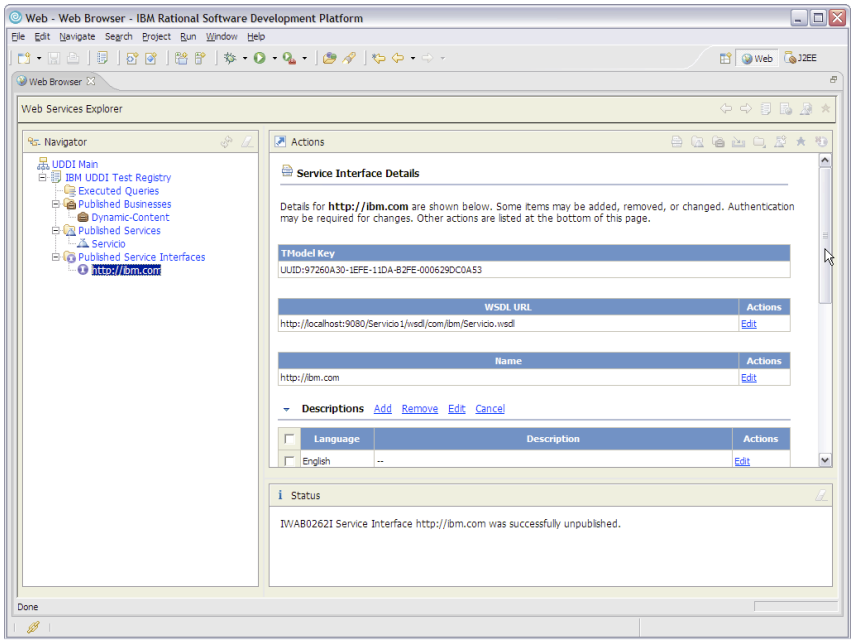


Escriba el nombre de su negocio y una corta descripción de lo que hace. Incluya los datos de su cuenta (usuario y password) en el registro UDDI.



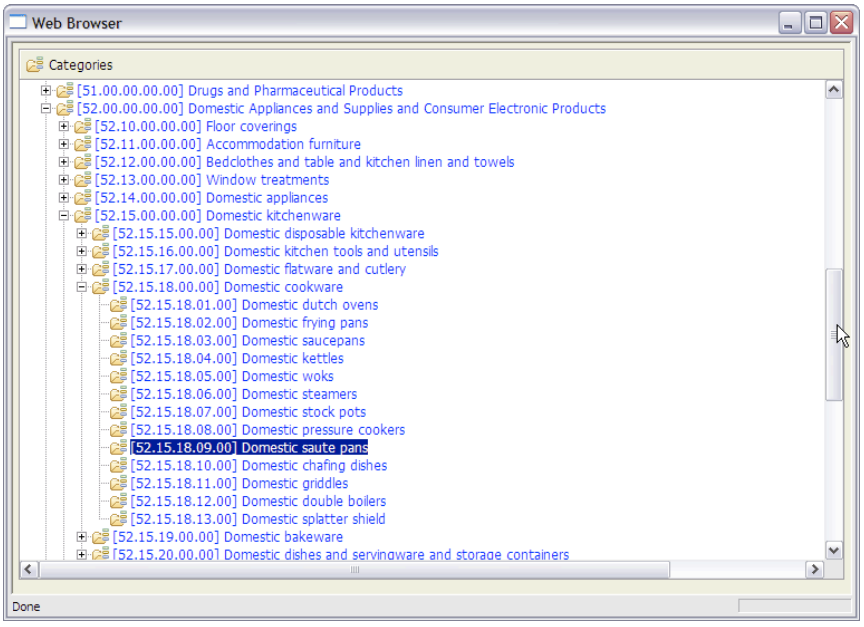
Pulse el botón “Go” para registrar el negocio. En la ventana de “Status” aparecen mensajes que indican si la operación se ha ejecutado normalmente.

A continuación aparece una pantalla que permite complementar los datos registrados del negocio con descripciones de su misión en varios idiomas y clasificarlo adecuadamente.

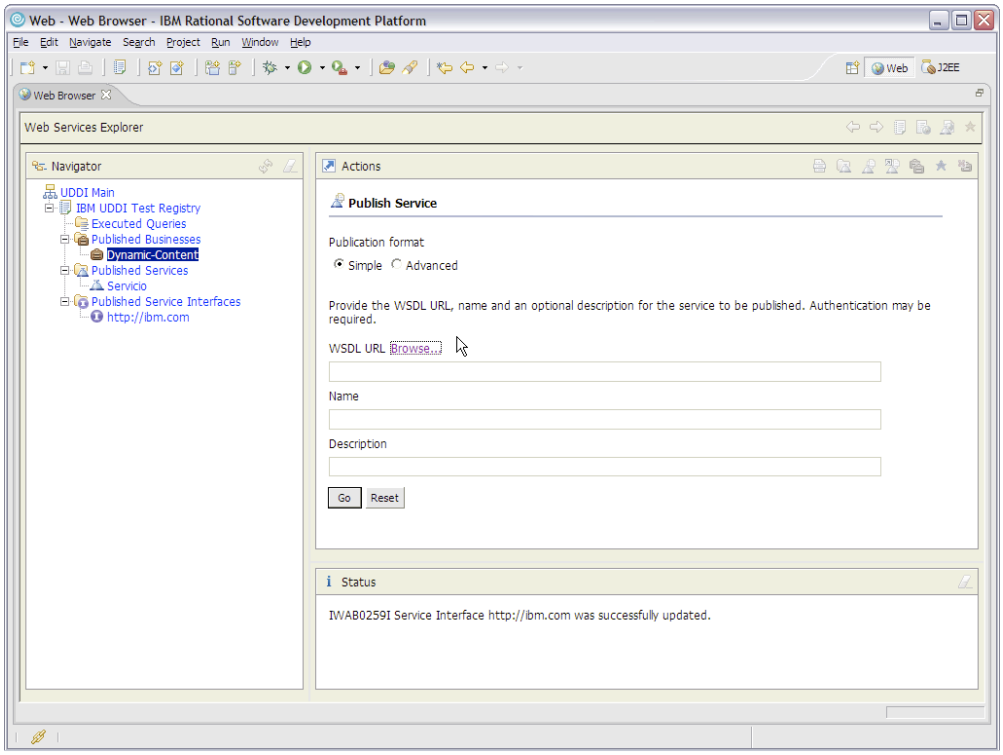


Existen varias maneras de clasificar los negocios. Uno de los sistemas más utilizados es el de la ONU (UNSPSC 7.3 United Nations Standard Products and Service Classification) pero existen otros como por ejemplo NAICS (North American Industrial Classification System). Una empresa puede ofrecer múltiples productos y por lo tanto en la sección Categories es posible agregar toda una lista de tipos.

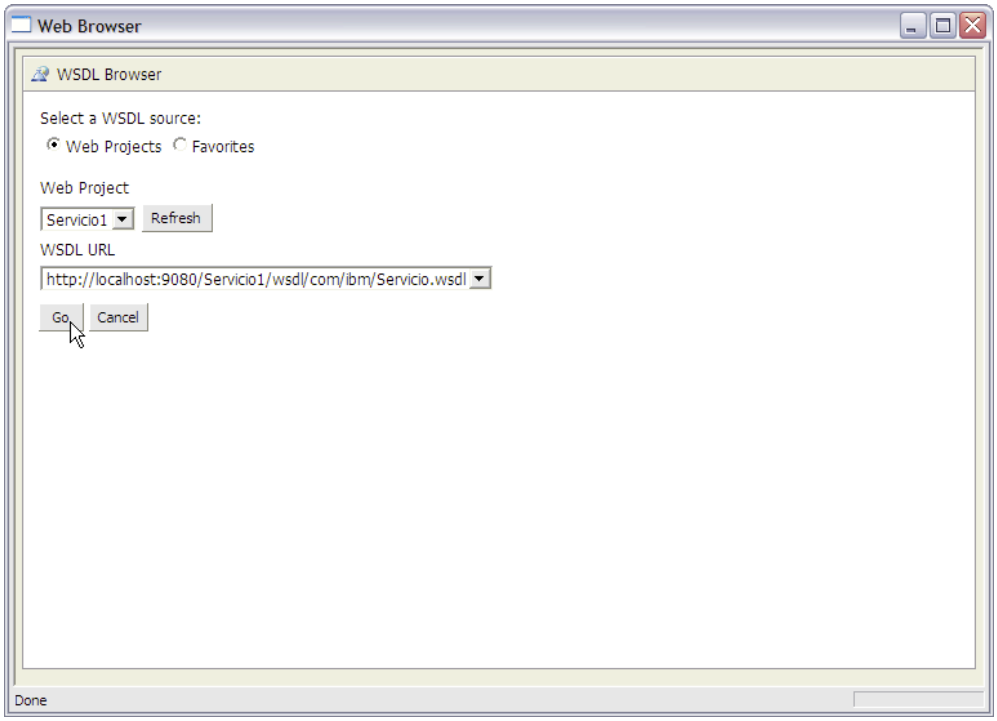
Supongamos que nuestra empresa producee artículos de cocina. Podemos hacer click en la liga “Add” junto a “Categories”. Seleccione como tipo UNSPC 7.3 y a continuación haga click sobre la ligra “Browse”. Esto abre una ventana de la cual puede elegir la categoría correcta, por ejemplo “52.15.18.09.00 Domestic saute pans”.



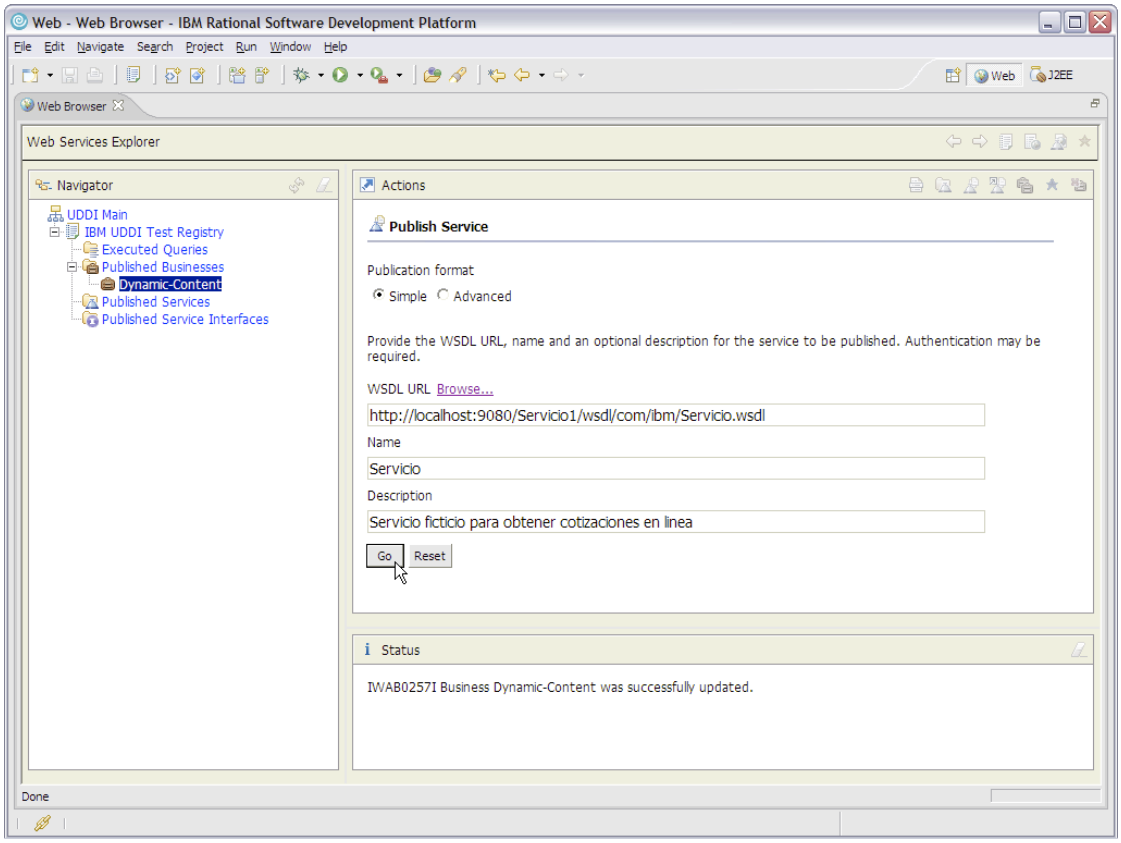
Pulse “Go” para guardar los cambios. Hora ya está llisto para publicar el servicio.



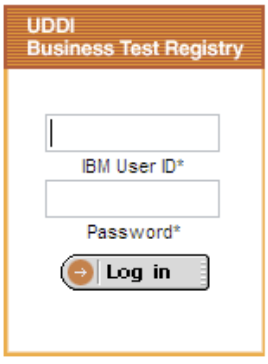
Pulse la liga “Browse” para seleccionar el servicio que desea publicar. Como solo tenemos un servicio, no necesitamos completar ninguna información. Tenga en cuenta que para que esta pantalla aparezca como se muestra en la ilustración el servidor de aplicaciones debe estar arrancado. Si no lo estuviera, arránquelo y cierre la ventana antes de volver a intentarlo.



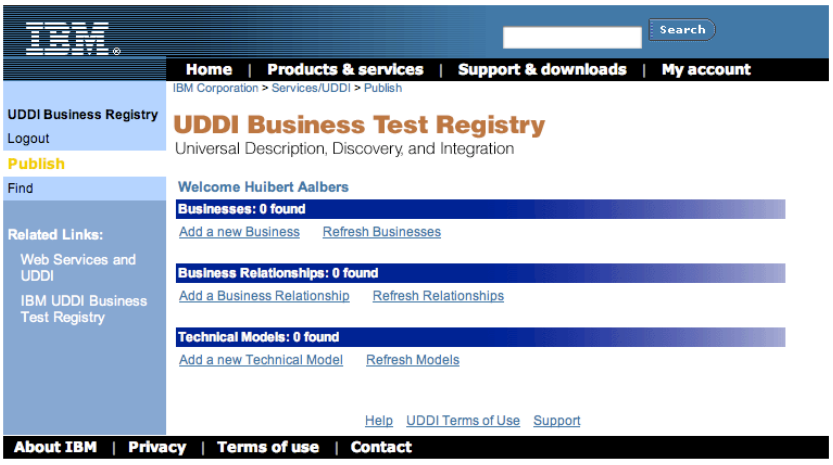
Complete los datos faltantes (Nombre y Descripción) y pulse “Go” para registrar los cambios.



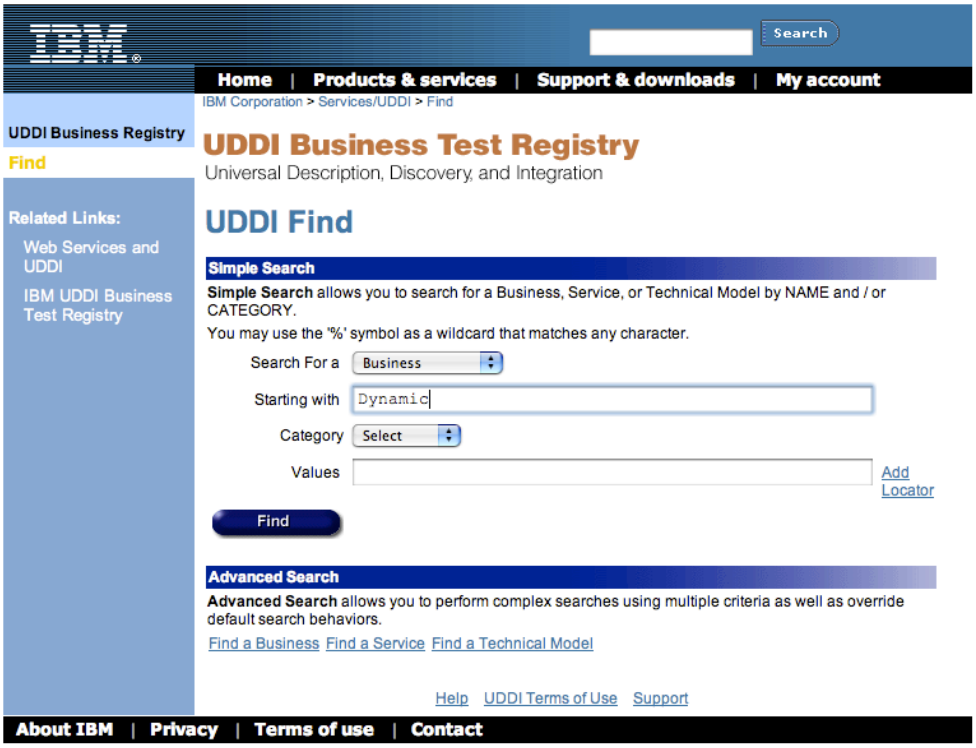
Ya está, eso es todo. El servicio está publicado en el registro público de IBM y puede ser consultado por cualquier persona interesada por Intenet. Comprobemos que eso es cierto conectándonos al registro a través del interfaz web mediante un browser.



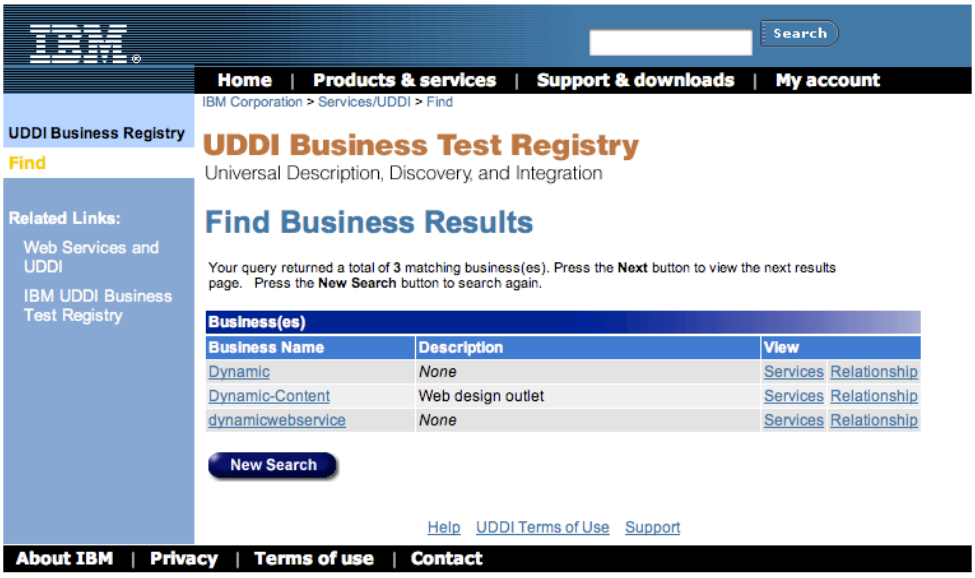
Regístrese con su nuevo usuario. El sistema le da la bienvenida y le da acceso al interfaz web del directorio UDDI tal y como se muestra en la siguiente ilustración.



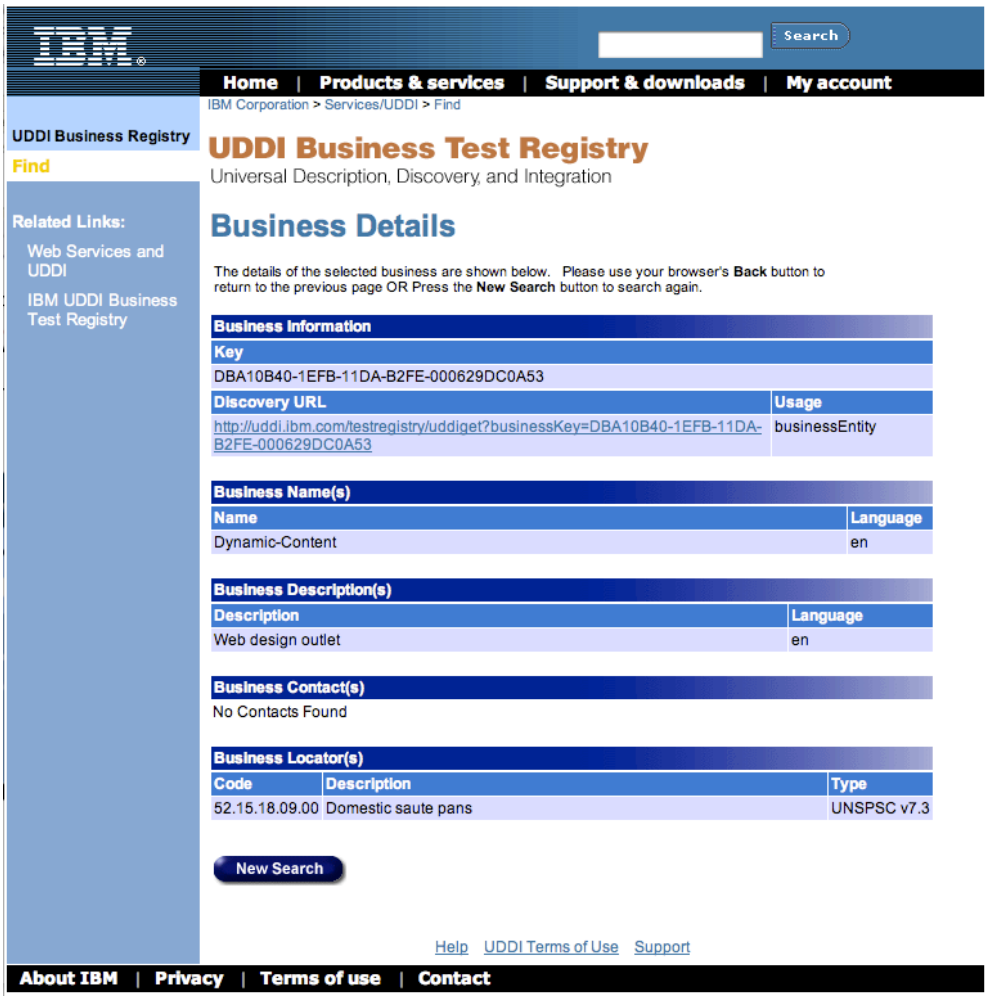
Vamos a realizar una búsqueda del negocio que acabamos de publicar para verificar que la actualización se ha realizado correctamente.



Escriba las primeras letras del nombre del negocio que acaba de registrar y pulse el botón “Find”.



El directorio despliega las empresas que corresponden al criterio de búsqueda. Haga click sobre el nombre de la empresa que acaba de registrar para ver los detalles.



Antes de dar por terminado este laboratorio explore un poco más las funciones que provee un directorio UDDI.

Laboratorio 3
Creación de un cliente de un servicio web partiendo de un archivo WSDL

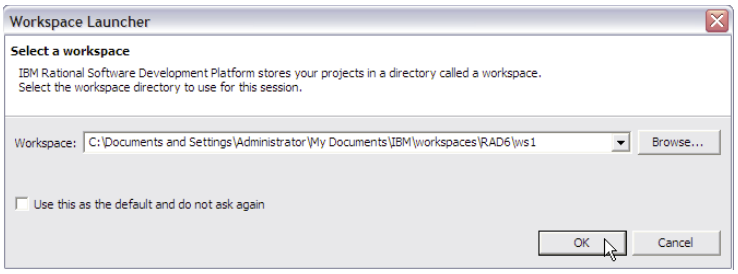
Objetivos
En este laboratorio vamos a crear un cliente para el servicio web creado en el laboratorio 1.

Prerrequisitos
Haber completado con éxito los laboratorios 1 y 2.

Instrucciones

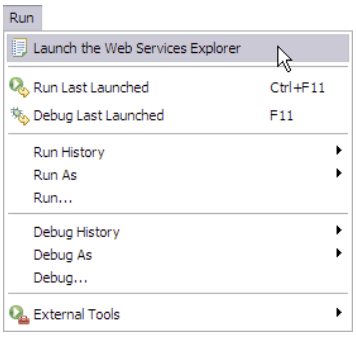
Inicie el entorno de desarrollo pulsando el siguiente icono en la barra de tareas  .

Elija el workspace creado en el laboratorio anterior.



Lo primero que vamos a hacer es crear un nuevo proyecto web para albergar el cliente del servicio. Es importante para los efectos de este laboratorio que el cliente y el servicio estén en distintos proyectos de web porque eso nos asegura que el cliente funciona incluso corriendo en otro equipo.

Haga right-click sobre “Dynamic Web Projects” y seleccione New >> Dynamic Web Project del menú contextual. Nombre “Cliente” a ese nuevo proyecto. Pulse el botón “Finish” para terminar.

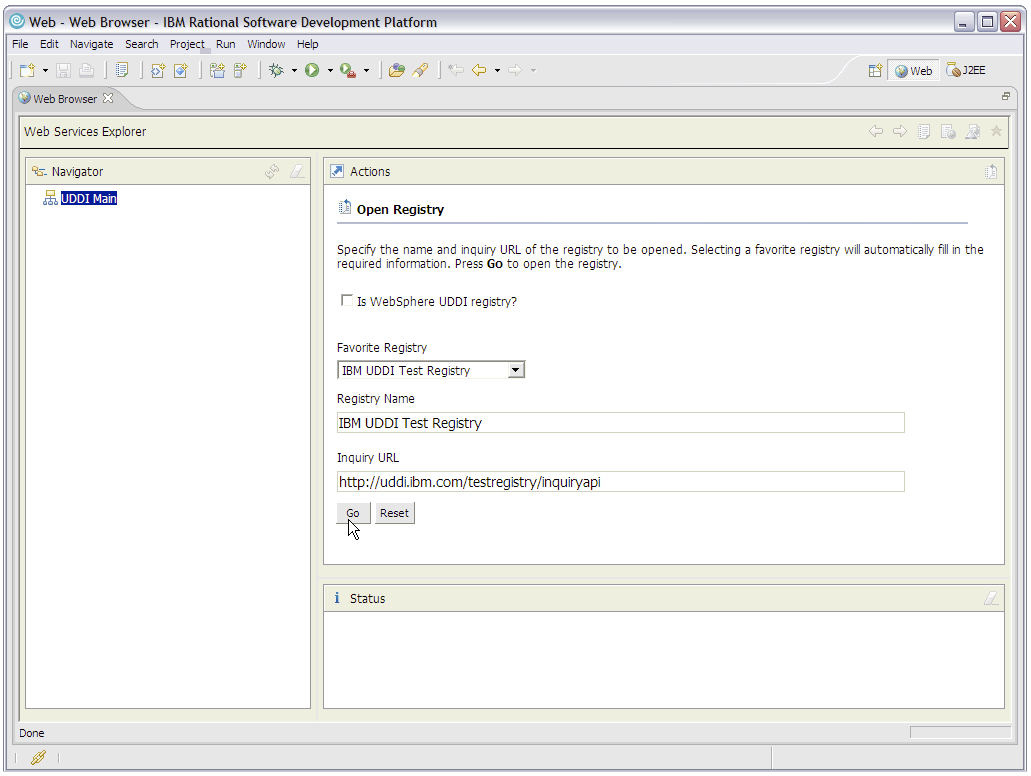


En este laboratorio vamos a utilizar el archivo WSDL que publicamos en el directorio UDDI en el laboratorio anterior para generar el código cliente necesario para invocar el servicio web desde una página JSP.

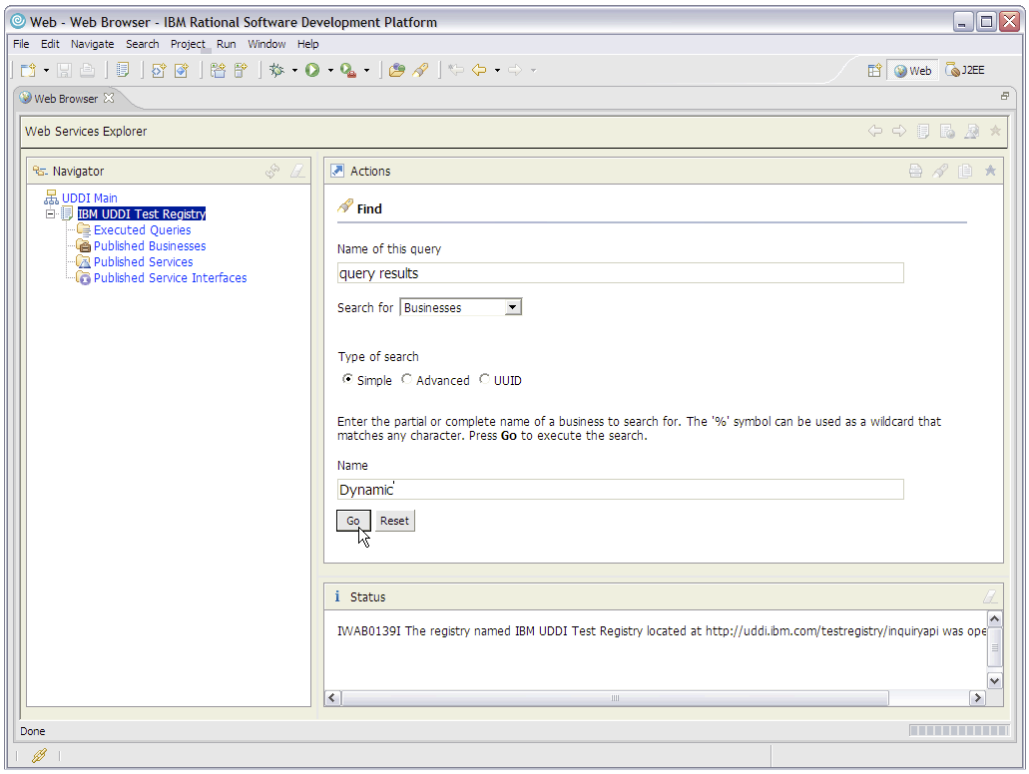
Lo primero que tenemos que hacer es obtener el archivo WSDL del directorio UDDI utilizando el Web Services Explorer que nos proporciona Rational Application Developer. Para abrirlo seleccionamos la opción “Launch the Web Services Explorer” del mnú Run.

Esto tiene como efecto abrir el Web Services Explorer. Sin embargo, no podemos hacer nada hasta no conectarnos al directorio de pruebas de IBM.

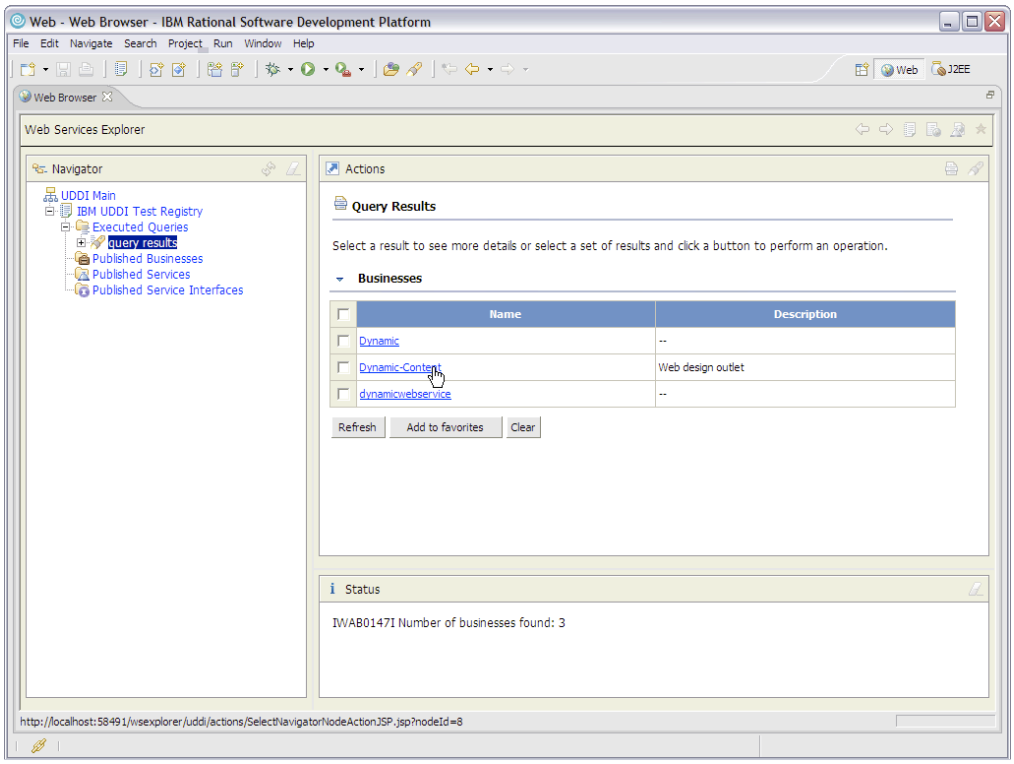
Maximice la ventana haciendo doble-click sobre la pestaña “Web Browser”. Asegúrese que el checkbox “Is WebSphere UDDI registry” no está tachado y seleccione el “IBM UDDI Test Registry” en el menú pull-down. Pulse el botón “Ok” para continuar tal y como se muestra en la siguiente illustración.

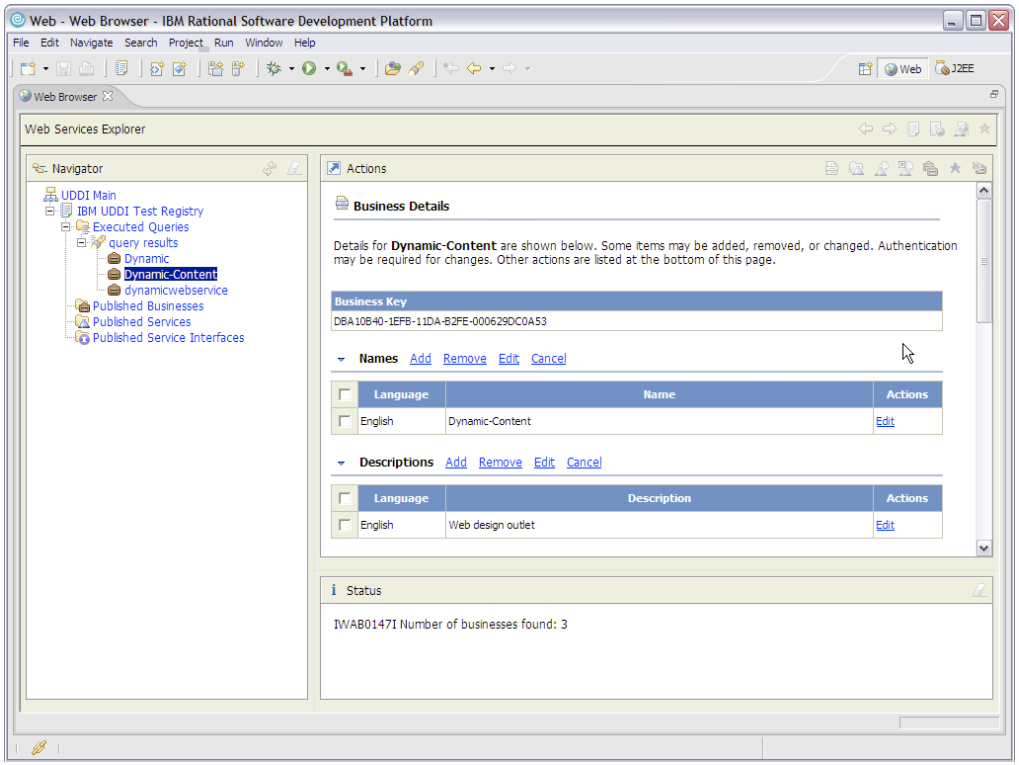


Ahora ya estamos conectados al directorio UDDI de pruebas de IBM. Ahora tenemos que buscar el negocio que creamos en el laboratorio principal.

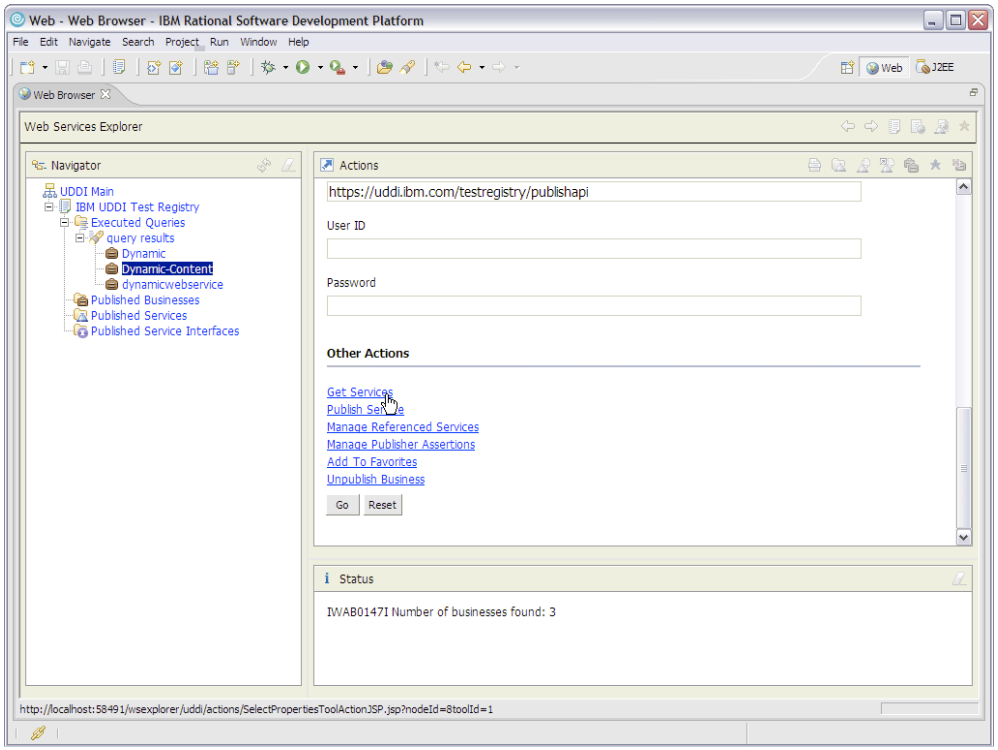


Escriba las primeras letras del nombre de la empresa que creó anteriormente en el campo “Name” y luego pulse el botón “Go”. La búsqueda puede arrojar varios resultados, tal y como se muestra a continuación. Haga click sobre el nombre de la empresa correcta para obtener la información completa.

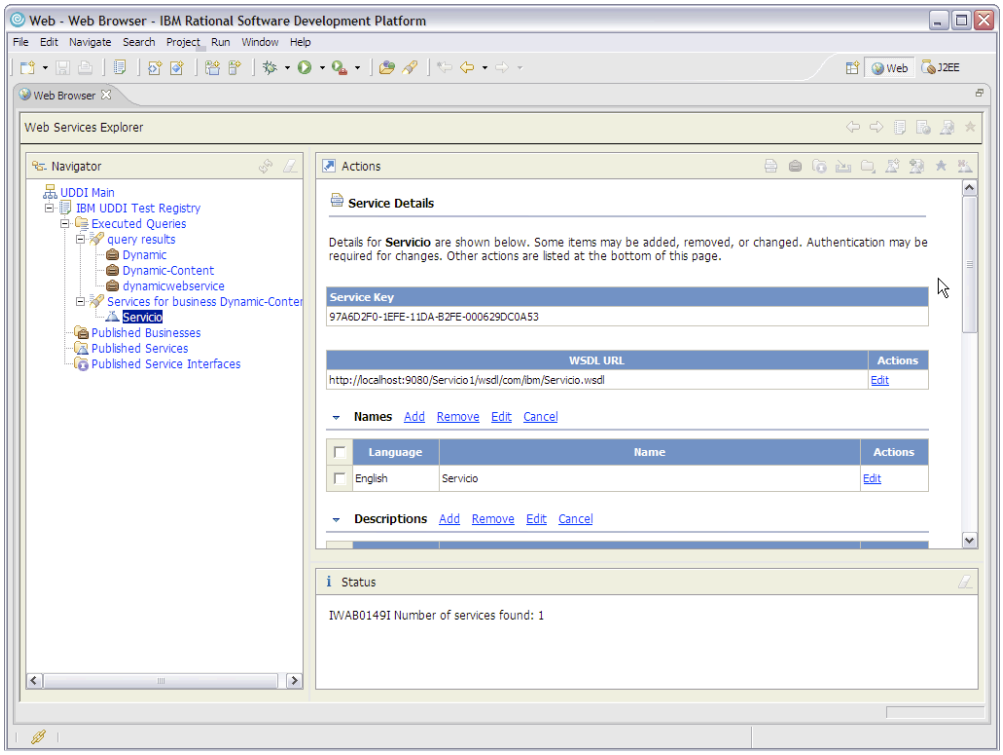




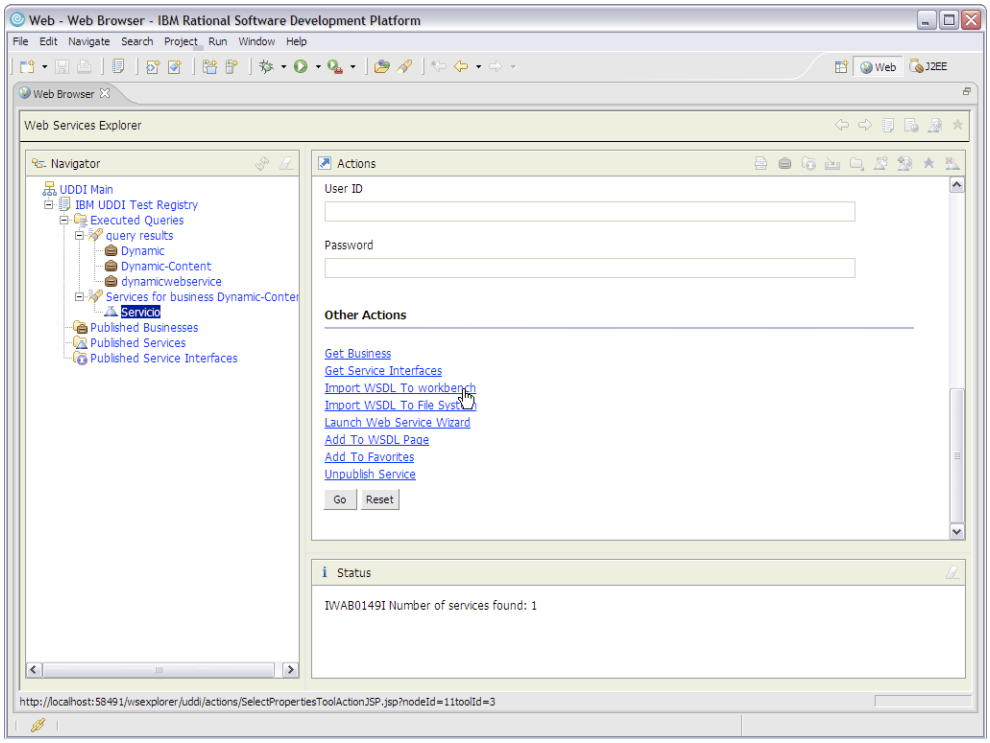
Si baja hasta el final de la página verá una serie de ligas. Haga click sobre “Get Services para cargar dentro del Web Services Explorer la definición de los servicios que tiene definidas la empresa.



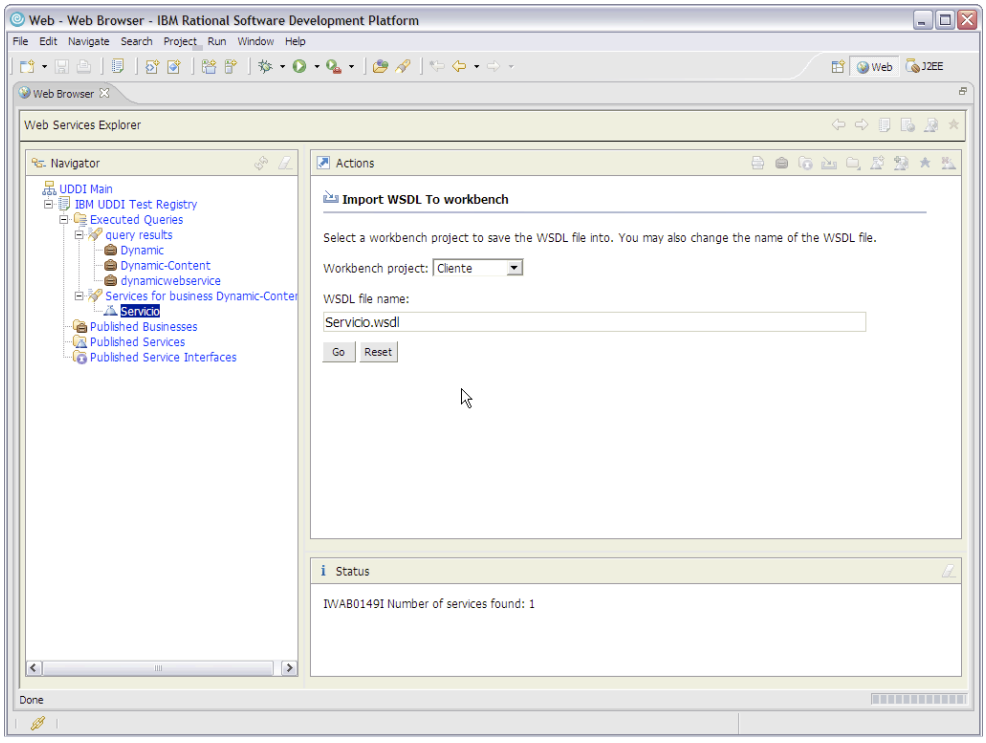
Ahora podemos ver en la parte de la derecha que el “Web Services Explorer” ya ha cargado la definición del servicio “Servicio”. Haga click sobre ese servicio.



Haga scroll hasta alcanzar la parte inferior de la ventana central. Haga click sobre la opción “Import WSDL To workbench:”.



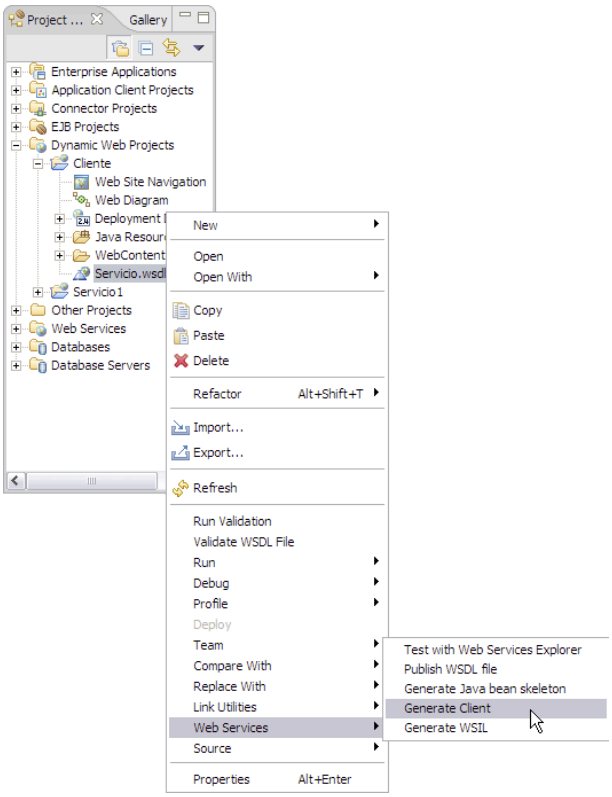
Ahora el prohrama nos gará unas preguntas para determinar con exactitud dónde debe colocar el archivo WSDL.



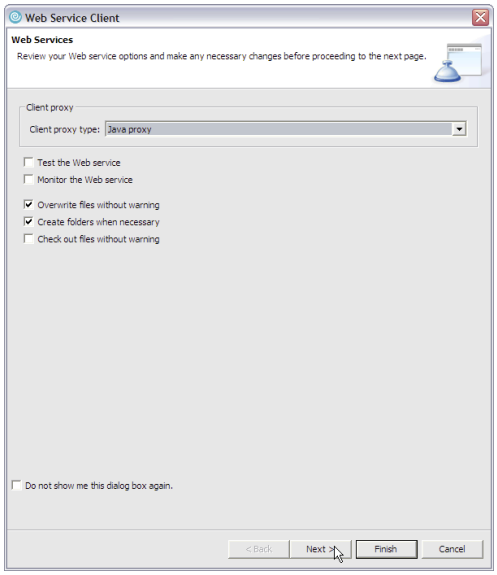
Indique que el archivo WSDL debe ser copiado al proyecto “Cliente”. No cambie el nombre del servicio. Pulse el botón “Go” para continuar.

El archivo WSDL está ahora en el proyecto “Cliente”. Ciertamente, hubiéramos podido lograr lo mimo simplemente copiando el archivo del proyecto “Servicio1” al proyecto “Cliente” pero el objetivo de este laboratorio es mostrar cómo deberían proceder para generar un cliente creado por otra organización y publicado en un directorio UDDI.

Ahora vamos a crear el cliente Java para el servicio a partir del archivo WSDL que acabamos de obtener.

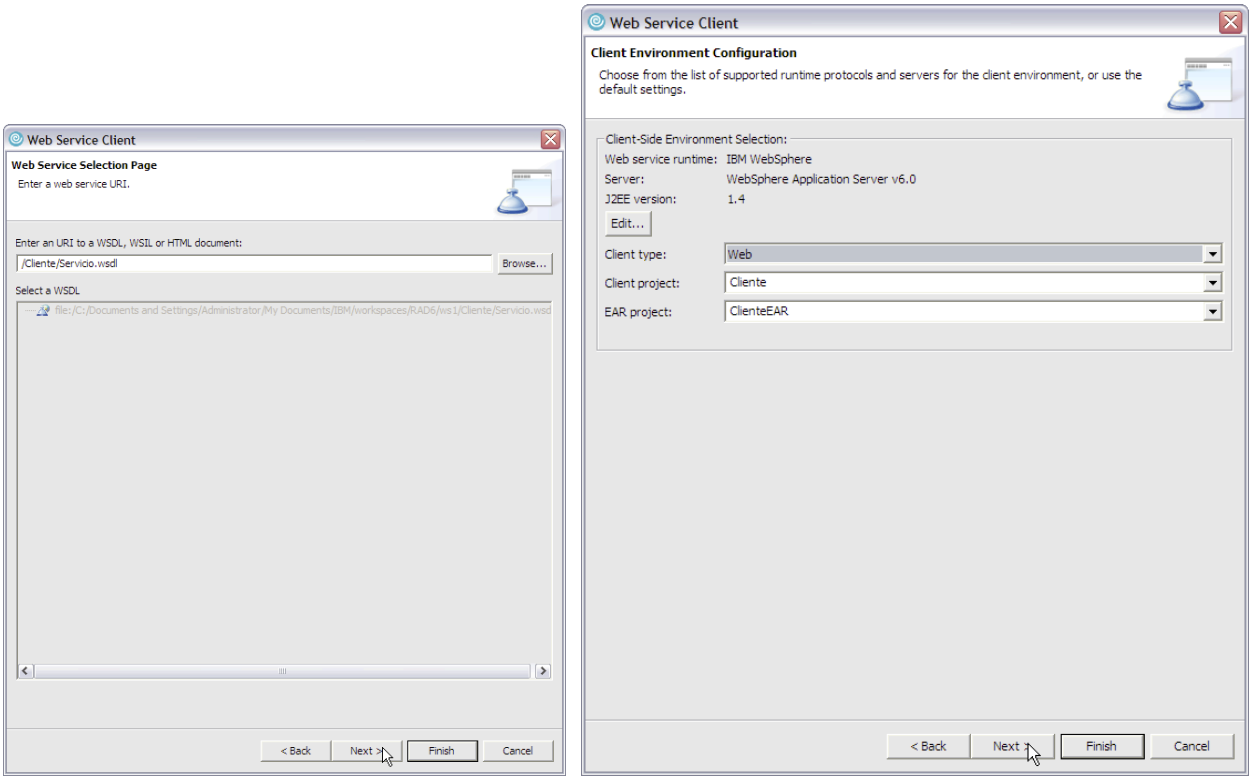


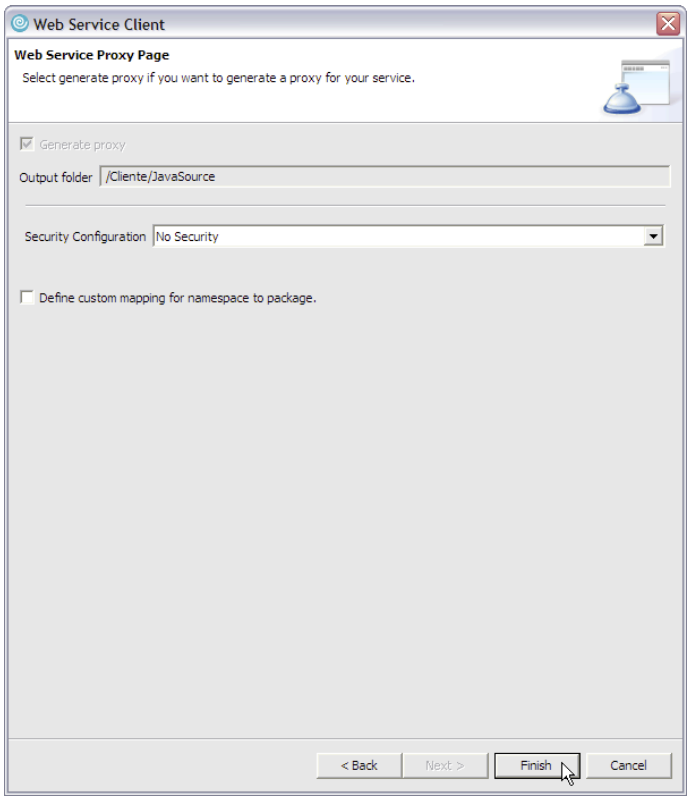
Haga right-click sobre el archivo WSDL y seleccione la opción Web Services >> Generate Client. Este asistente se va a encargar de generar las clases necesarias para poder invocat fácilmente el servicio web que desarrollamos en el laboratorio 1. Sin embargo, para ligrarlo primero tiene que contestar varias preguntas.



Pulse simplemente “Next” para continuar.

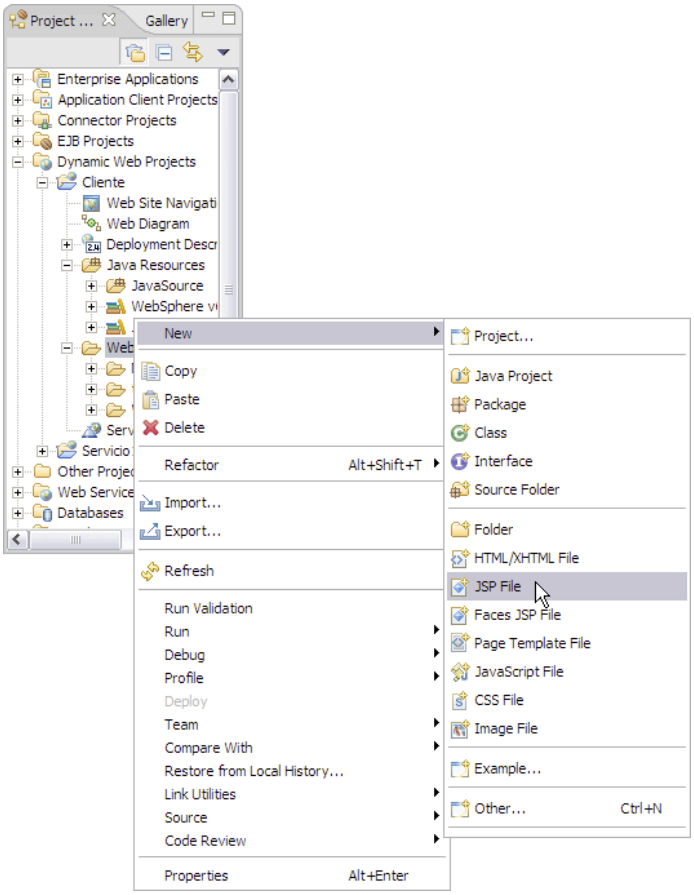
En principio tampoco es necesario cambiar ningún valor en las siguientes pantallas, pero verifique los valores por defecto para entender lo que está haciendo el asistente. Pase de una pantalla a la siguiente usando el botón “Next >” y finalmente el botón “Finish” en la última pantalla del asistente para terminar.





El asistente genera de manera automática todas las clases necesarias para invocar el servicio y coloca todas esas clases en el directorio Java Resources/Java Source..

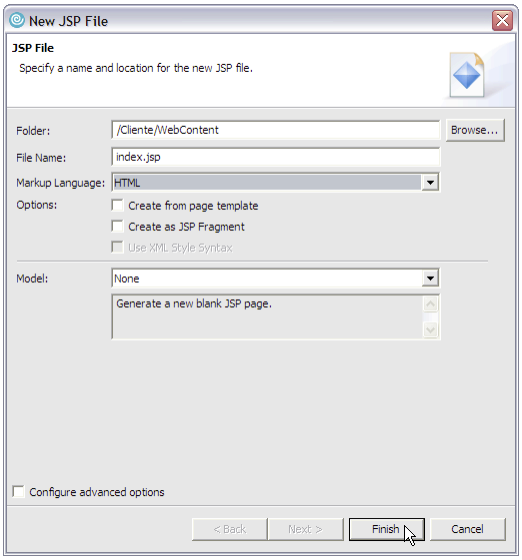
Finalmente, para terminar solo nos falta crear la página JSP que vamos a usar para invocar el servicio.



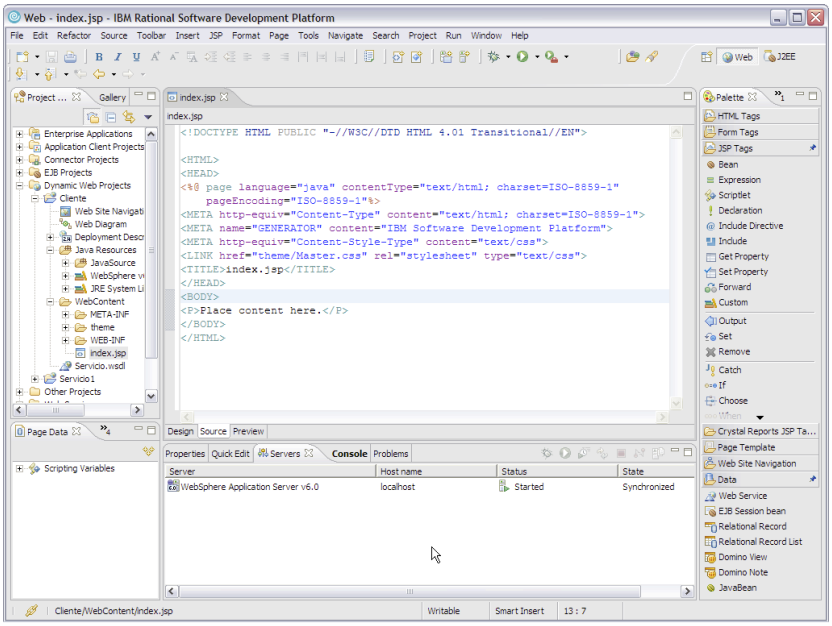
Pata ello vamos a usar el asistente que provee Rational Application Developer v. 6.0.

Haga right-click sobre el directorio WebContent. Seleccione la opción New >> JSP File del menú contextual.

Nombre la nueva página index.jsp y pilse el botón “Finish” para terminar.



Esto va a crear la nueva página y va a desplegarla en un editor gráfico. Haga click sobre el tab de “Source” para trabajar spbre el código fuente de la página.



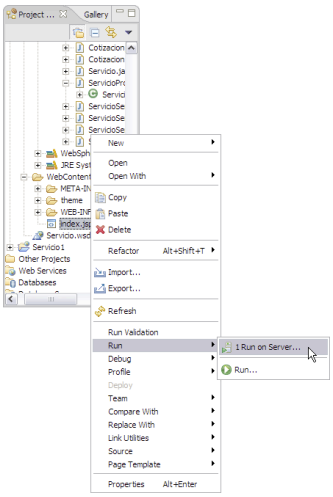
Copie el siguiente código en su página:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ page import="com.ibm.*"%>

<HTML>
<HEAD>
<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<META name="GENERATOR" content="IBM Software Development Platform">
<META http-equiv="Content-Style-Type" content="text/css">
<TITLE>index.jsp</TITLE>
</HEAD>
<BODY>
<H3>Cliente del servicio obtenerCotizacion().</h3>
<%
   float value = 0;

   if (request.getParameter("simbolo") == null){
%>
      <form name="miForma" method="POST" action="index.jsp">
         <TABLE border="3" cellspacing="2" cellpadding="2">
            <tr>
               <th>S&iacute;mbolo</th>
               <td><input type="text" name="simbolo"></td>
               <td><input type="text" name="valor"></td>
               <td><input type="submit" value="Consultar"></td>
            </tr>
         </TABLE>
      </form>
<%
   }
   else
   {
      ServicioProxy proxy = new ServicioProxy();
      Servicio servicio = proxy.getServicio();

      try {
         value =
servicio.obtenerCotizacion(request.getParameter("simbolo")).getValor();
%>
         <form name="miForma" method="POST" action="index.jsp">
            <TABLE border="3" cellspacing="2" cellpadding="2">
               <tr>
                  <th>S&iacute;mbolo</th>
                  <td><input type="text" name="simbolo"
value="<%=request.getParameter("simbolo")%>"></td>
                  <td><input type="text" name="valor" value="<%=value%>"></
td>
                  <td><input type="submit" value="Consultar"></td>
               </tr>
            </TABLE>
```

```
         </form>
<%
      }
      catch (Exception ex){
%>
         <form name="miForma" method="POST" action="index.jsp">
            <TABLE border="3" cellspacing="2" cellpadding="2">
               <tr>
                  <th>S&iacute;mbolo</th>
                  <td><input type="text" name="simbolo"></td>
                  <td><input type="text" name="valor"></td>
                  <td><input type="submit" value="Consultar"></td>
               </tr>
               <tr>
                  <th colspan="4">Error de comunicacioacute;n con el
servicio</th>
               </tr>
            </TABLE>
         </form>
<%
      }
   }
%>
</BODY>
</HTML>
```

El código importante es el que manda llamar el servicio web, en este caso

```
ServicioProxy proxy = new ServicioProxy();
Servicio servicio = proxy.getServicio();

try {
   value = servicio.obtenerCotizacion("IBM").getValor();
}
catch (Exception ex){
}
```



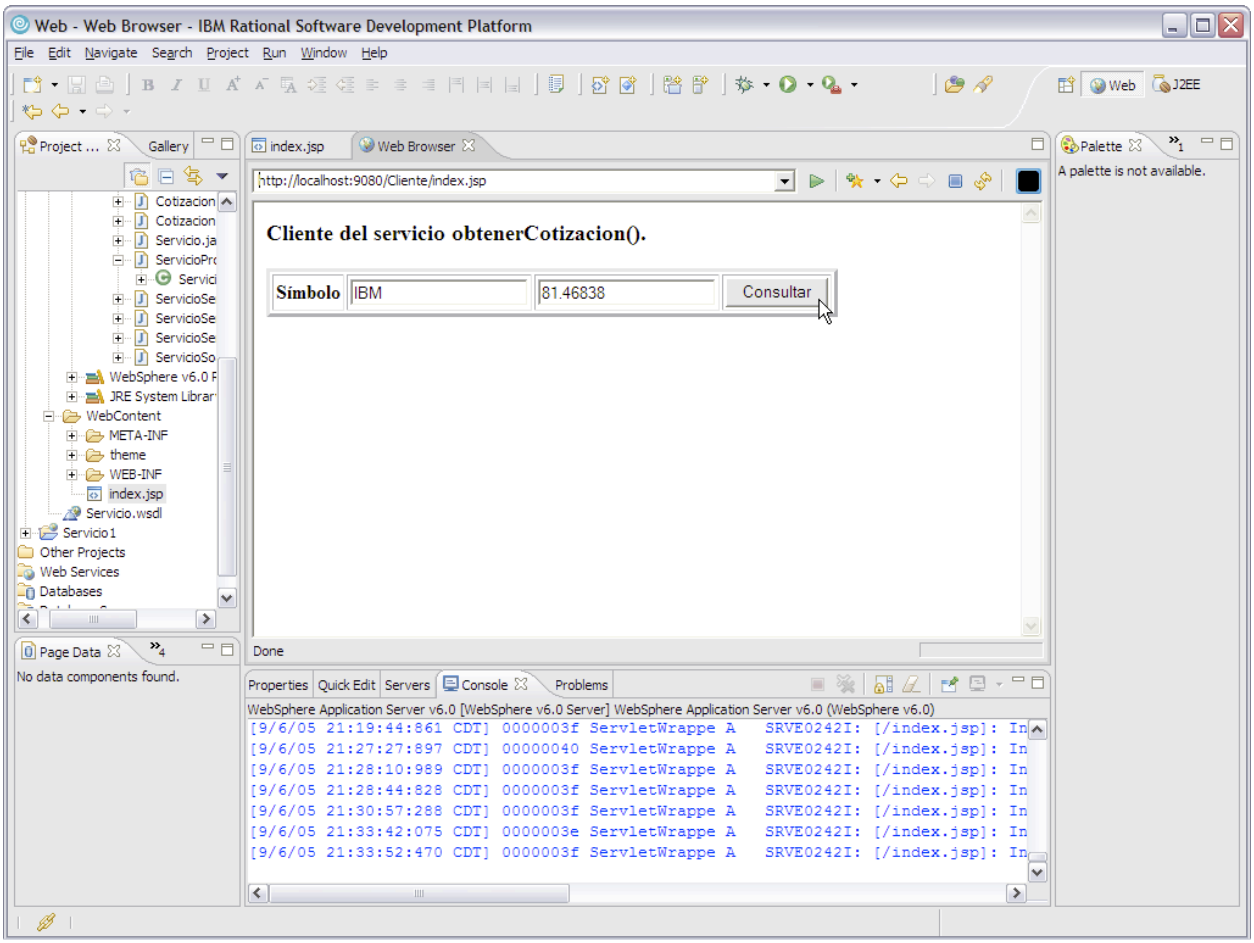
Como puede apreciar, invocar un servicio web es realmente muy sencillo, aunque es posible simplificar el proceso aún más como lo veremos en un siguiente laboratorio usando JSFs.

Salve los cambios en el archivo “index.jsp” con Ctrl-S.

Para probar la página solo es necesario hacer right-click sobre el nombre del archivo, en este caso “index.jsp” y seleccionar la opción Run >> 1 Run on Server...

En la siguiente ventana simplemente confirme que los datos por defecto son correctos y pulse “Finish”/ Si toso funciona correctamente, se abrirá un browser en el que podrá probar la aplicación tal y como se muestra en la última ilustración de este laboratorio.

¡Felicidades ha terminado con éxito los tres primeros laboratorios!


Web - Web Browser - IBM Rational Software Development Platform
File Edit Navigate Search Project Run Window Help
Web
J2EE
Project ...
Gallery
index.jsp
Web Browser
http://localhost:9080/Cliente/index.jsp
Cliente del servicio obtenerCotizacion().
Símbolo
IBM
81.46838
Consultar
Done
Palette
A palette is not available.
Page Data
No data components found.
Properties
Quick Edit
Servers
Console
Problems
WebSphere Application Server v6.0 [WebSphere v6.0 Server] WebSphere Application Server v6.0 (WebSphere v6.0)
[9/6/05 21:19:44:861 CDT] 0000003f ServletWrappe A SRVE0242I: [/index.jsp]: In
[9/6/05 21:27:27:897 CDT] 00000040 ServletWrappe A SRVE0242I: [/index.jsp]: In
[9/6/05 21:28:10:989 CDT] 0000003f ServletWrappe A SRVE0242I: [/index.jsp]: In
[9/6/05 21:28:44:828 CDT] 0000003f ServletWrappe A SRVE0242I: [/index.jsp]: In
[9/6/05 21:30:57:288 CDT] 0000003f ServletWrappe A SRVE0242I: [/index.jsp]: In
[9/6/05 21:33:42:075 CDT] 0000003e ServletWrappe A SRVE0242I: [/index.jsp]: In
[9/6/05 21:33:52:470 CDT] 0000003f ServletWrappe A SRVE0242I: [/index.jsp]: In
Cotizacion
Cotizacion
Servicio.ja
ServicioPr
Servici
ServicioSe
ServicioSe
ServicioSe
ServicioSo
WebSphere v6.0 F
JRE System Librar
WebContent
META-INF
theme
WEB-INF
index.jsp
Servicio.wsdl
Servicio1
Other Projects
Web Services
Databases

Laboratorio 4
Creación de un cliente web utilizando Java Server Faces (JSF)

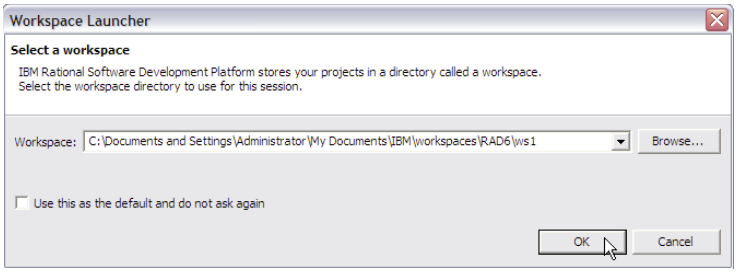
Objetivos
En el laboratorio anterior se demostró como programar un cliente web utilizando los APIs de Web Services que provee Java. A pesar de que se trata de algo sencillo, es posible simplificarlo aún más a través del uso de la tecnología de Java Server Faces que permite el uso de componentes reutilizables dentro de páginas JSP lo que simplifica el desarrollo y el mantenimiento del código.

Prerrequisitos
Rational Application Developer debe estar instalado en su equipo.

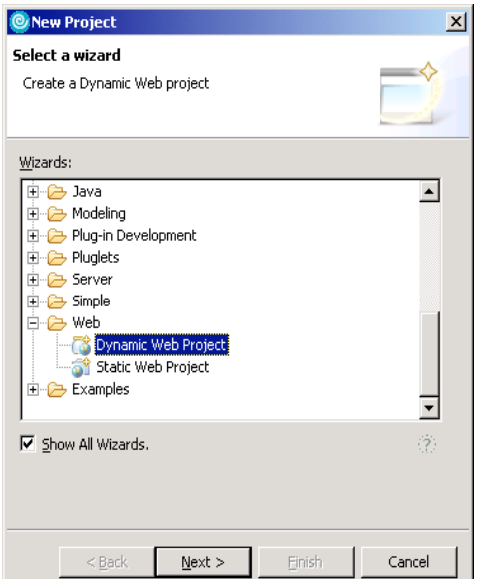
Instrucciones

Inicie el entorno de desarrollo pulsando el siguiente icono en la barra de tareas  .

Si ha realizado los laboratorios 1,2 y 3 elija el workspace creado anteriormente, de lo contrario cree un nuevo.

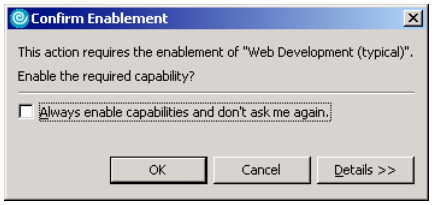


Lo primero que vamos a hacer es crear un nuevo proyecto web para albergar el cliente del servicio. Seleccione la opción File > New > Project en la barra de menús.

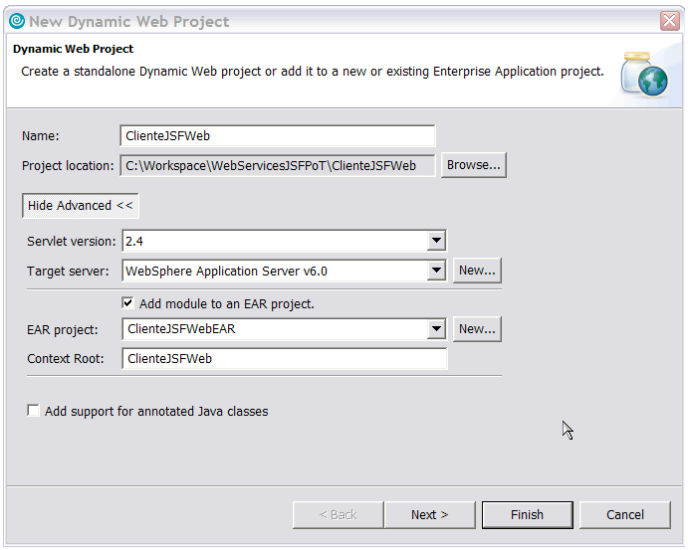


Seleccione “Dynamic Web Project” dentro de la carpeta Web y pulse el botón “Next >”.

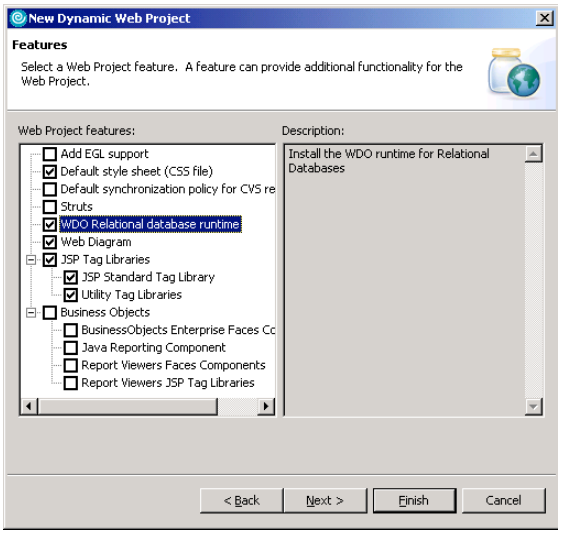
Si acaba de crear un nuevo workspace el sistema le va a preguntar si desea activar la funcionalidad de desarrollo web. Conteste de manera afirmativa pulsando el botón “OK”.



En la siguiente pantalla escriba el nombre de su proyecto, por ejemplo “ClienteJSFWeb” tal y como se muestra a continuación. Pulse el botón “Next >” para continuar.



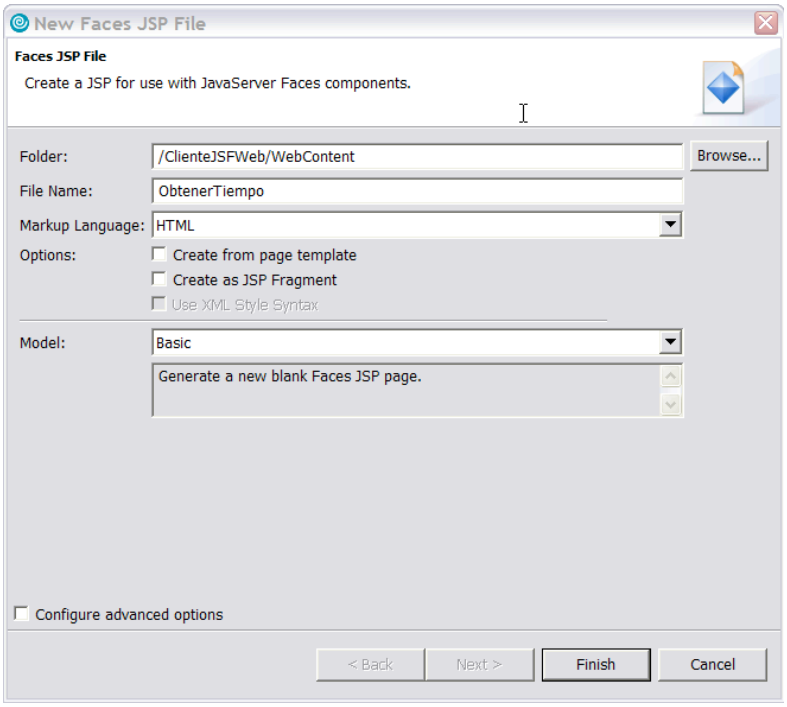
En la siguiente pantalla, asegúrese de que la opción “JSP Tag Libraries” está activada ya que Java Server Faces requiere de custom tags (en particular la JSP Standard Tag Library) para funcionar.

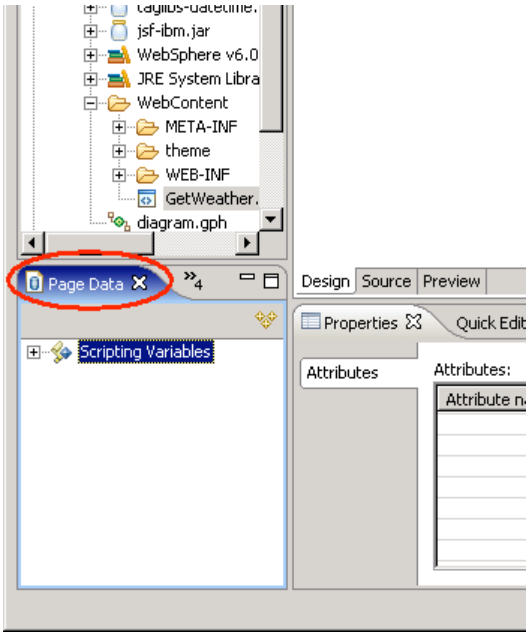


Pulse el botón “Finish” para terminar. El proyecto ya sido creado y estamos listos para empezar a crear el cliente web basado en JSF.

En la vista de Project Explorer, haga right-click sobre el nombre del proyecto web (ClienteJSFWeb) y seleccione la opción New->Faces JSP File del menú contextual.

Elija un nombre para su nueva página, por ejemplo “ObtenerStock” y pulse el botón “Finish” para terminar.





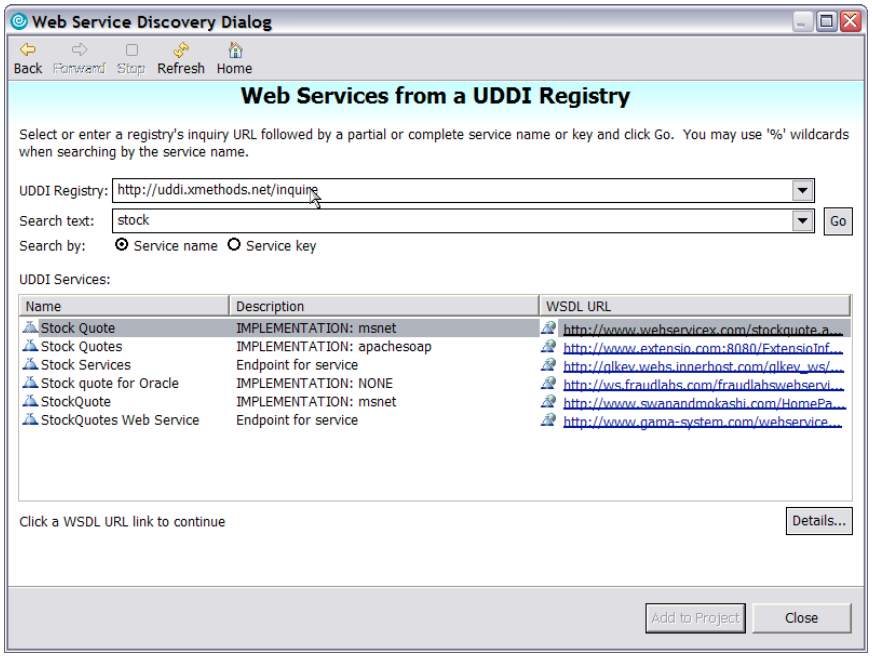
Ahora, en la vista de “Page Data”, la cual en la perspectiva de Web se encuentra situada en el ángulo inferior izquierdo de la pantalla, tal y como se muestra en la ilustración, haga right-click en el área vacía. Seleccione la opción New > Web Service del menú contextual.

Indique al asistente que desea accesar un web service público que ha sido publicado en un registro UDDI seleccionando la opción “Web Services from a UDDI Registry”.

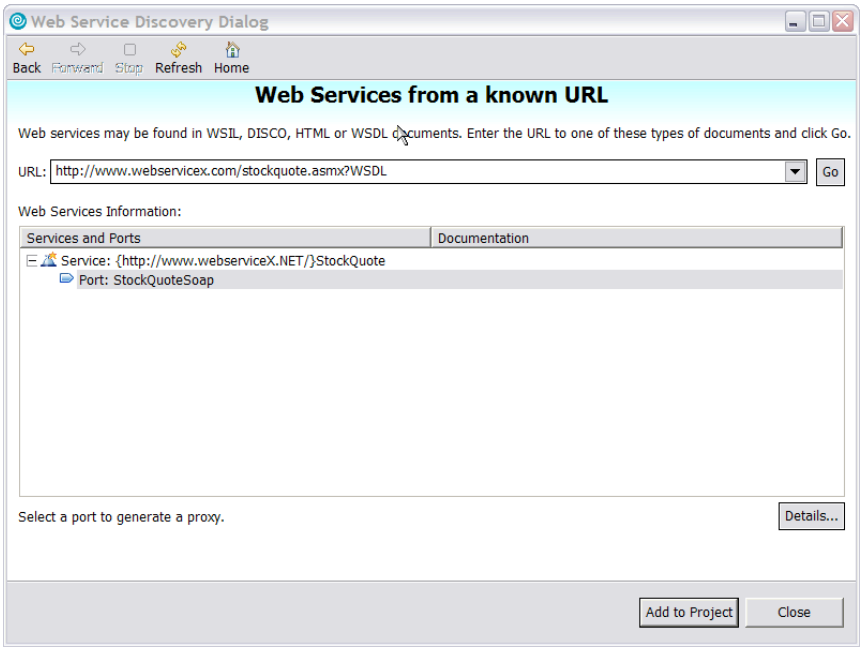
En la siguiente pantalla indique que desea utilizar el registro UDDI ubicado en http://uddi.xmethods.net/inquire y que desea desplegar todos los servicios que contengan la palabra “Stocks” tal y como se muestra en la siguiente captura de pantalla. Pulse el botón “Go” para iniciar la búsqueda.

Si todo funciona normalmente le deben aparecer varios resultados. Elija el servicio cuyo URL sea http://www.webservicex.com/stockquote.asmx?WSDL.

Observe que este servicio ha sido implementado utilizando la tecnología .Net de Microsoft. Esa es la razón por la que decidimos utilizar ese servicio en particular ya que nos va a permitir demostrar que los servicios web, al estar basadas en estándares abiertos, eliminan la complejidad de integración entre aplicaciones creadas con diferentes tecnologías.

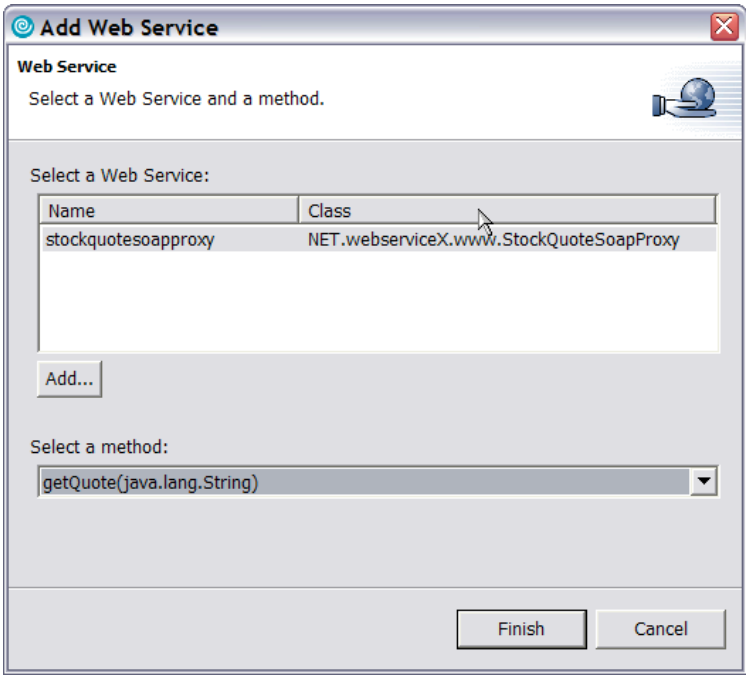


Haga click sobre el URL http://www.webservicex.com/stockquote.asmx?WSDL del archivo WSDL perteneciente al servicio “Stock quote”.



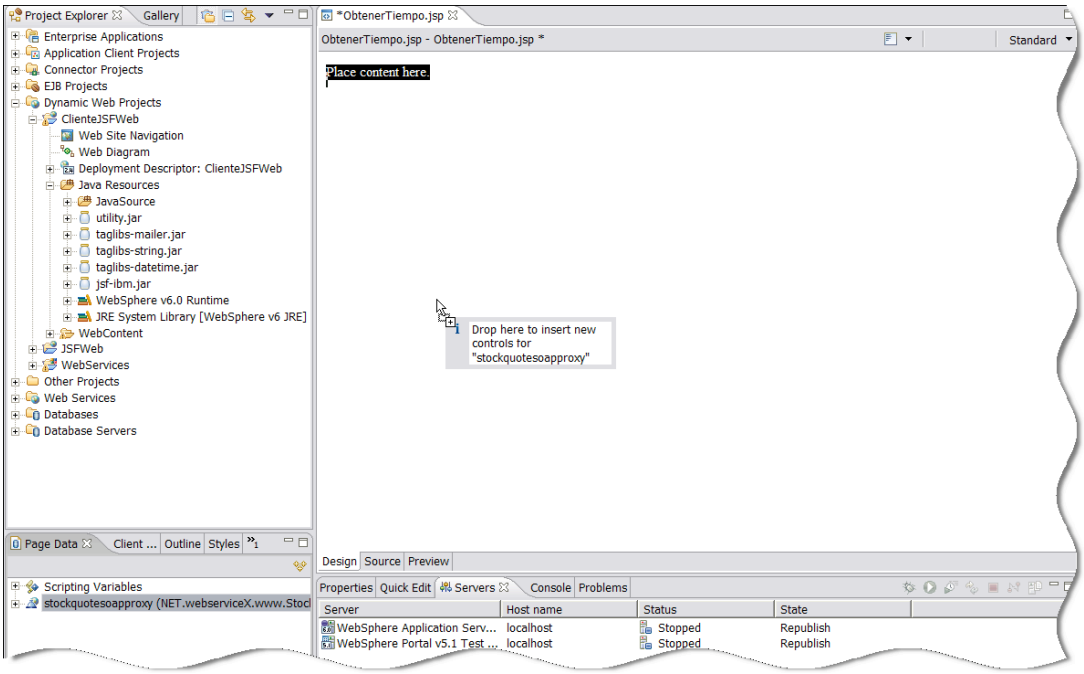
Después, marque la línea “Port: StockQuoteSoap…” y presione el botón “Add to Project”. Si aparece un warning pidiendo permiso para sobrescribir el archivo “web.xml”, responda “Yes to all”.

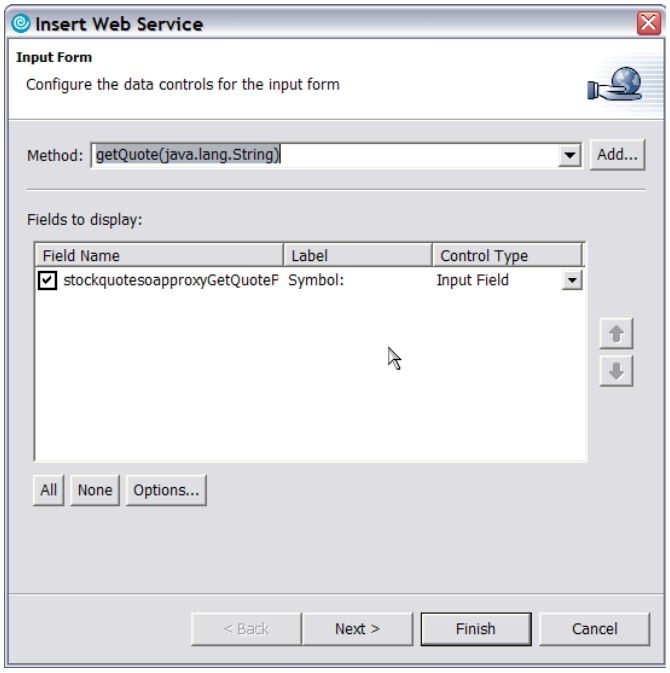
En la siguiente pantalla seleccione el método getQuote(java.lang.String) y pulse el botón “Finish” para terminar.



Verifique que en la vista “Page Data” haya aparecido un nuevo elemento llamado “stockquotesoapproxy”. Esto nos va a permitir utilizar fácilmente el servicio en las páginas de la aplicación.

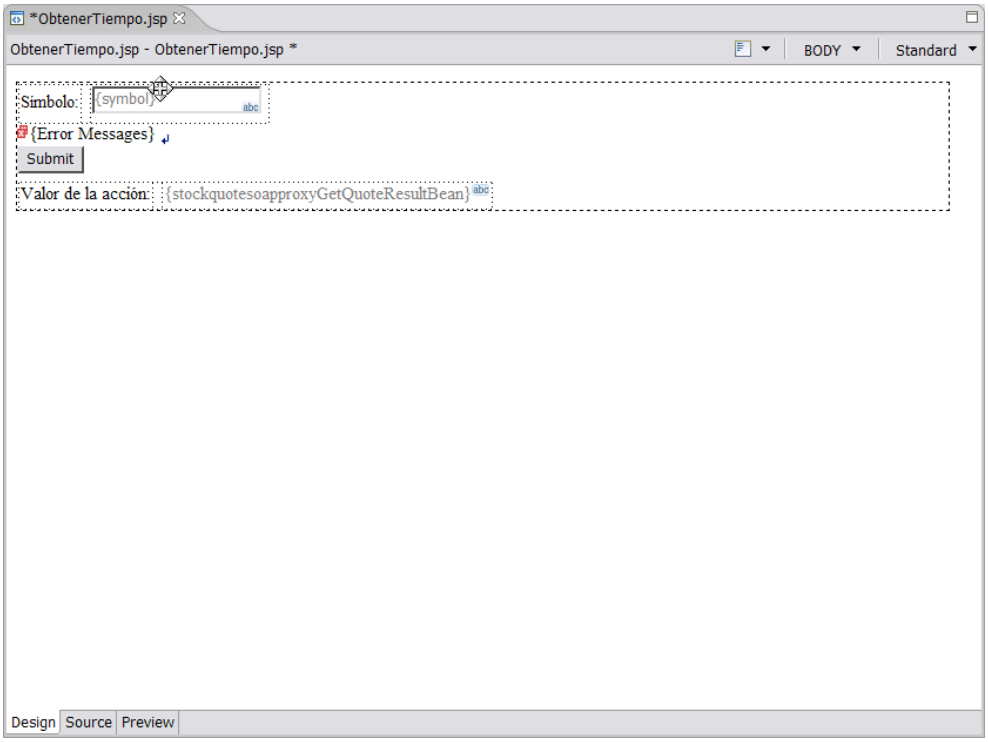
Para demostrarlo, arrastre simplemente el elemento “stockquotesoapproxy” sobre la página que creó anteriormente (ObtenerStock.jsp). Esto tiene por efecto arrancar el asistente “Insert Web Service” que permite configurar el comportamiento de la llamada al servicio web.



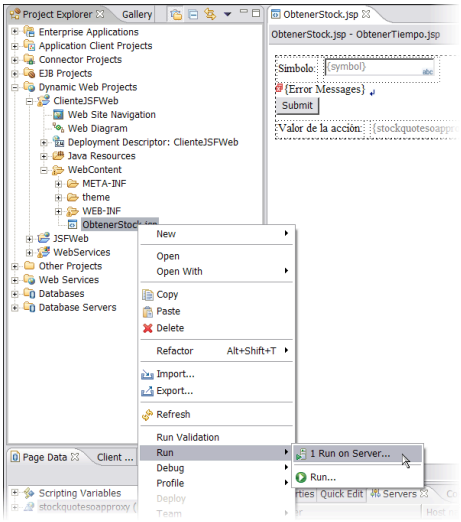


En este caso no vamos a realizar ningún cambio. Pulse el botón “Finish” para terminar.

Ahora ya solo nos falta arreglar un poco la página para que se vea mejor. Elimine el texto “Place content here.”. También cambie el texto “StockquotesoapproxyGetQuoteResultBean” por “Valor de la acción”.



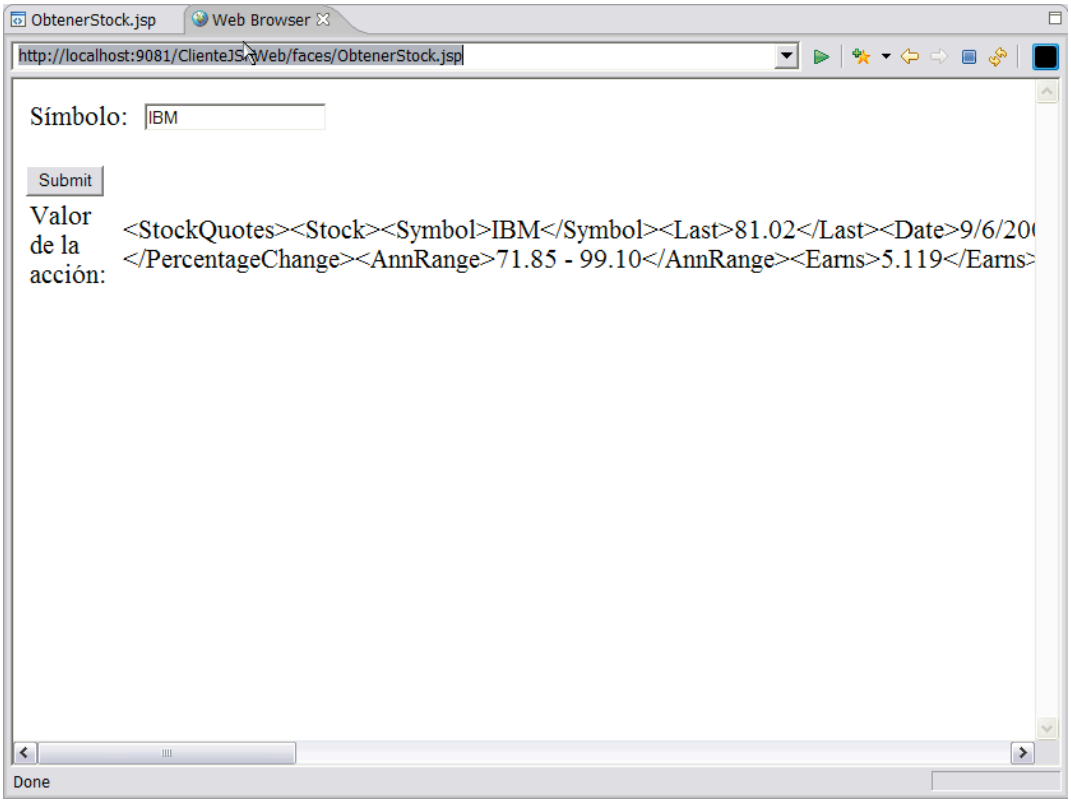
Para ejecutar la aplicación que acabamos de crear, simplemente haga right-click sobre el nombre de la página (ObtenerStock.jsp) dentro de la vista “Project Explorer” y seleccione la opción Run > Run on Server... del menú contextual.



Si aún no tiene un servidor definido, deberá definirlo ahora. El Apéndice 1 contiene las instrucciones necesarias para hacerlo. Si ya tiene definido un servidor, bastará con seleccionarlo para que esta aplicación sea instalada y arranque automáticamente.

Seleccione “WebSphere Application Server 6.0” y presione el botón “Finish”.

El servidor de aplicaciones tardará unos instantes en arrancar. Cuando termine, la página que acabamos de crear aparecerá en una nueva vista. Para probarla, ponga un símbolo bursátil (por ejemplo IBM o AAPL) en el campo correspondiente y pulse el botón “Submit”. El resultado debe aparecer tal y como se muestra en la siguiente ilustración.

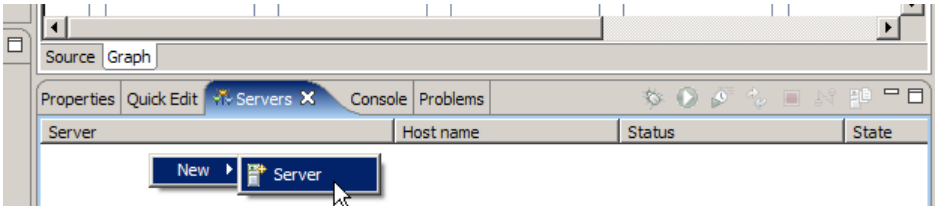


Como ejercicio adicional pueden intentar modificar la aplicación para que muestre solo el valor de la acción.

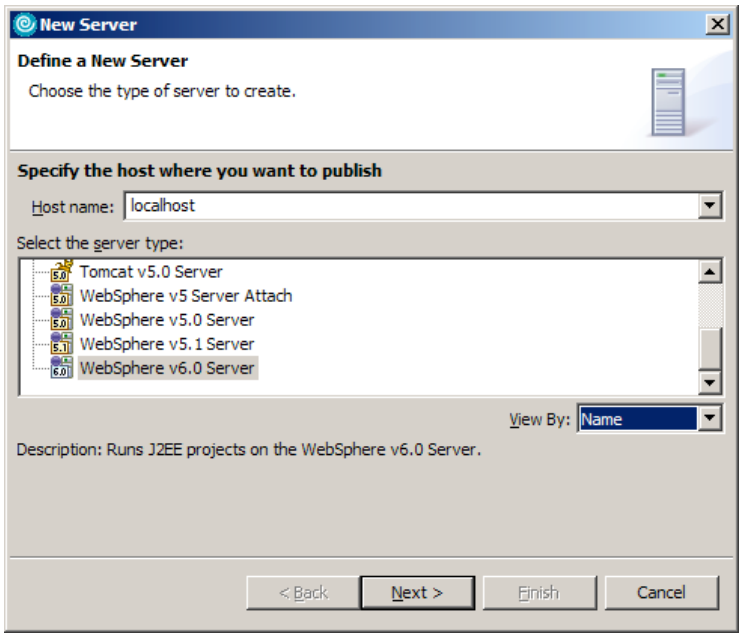
Apéndice 1
Configuración de un servidor

Esta tarea se realiza en tres sencillos pasos.

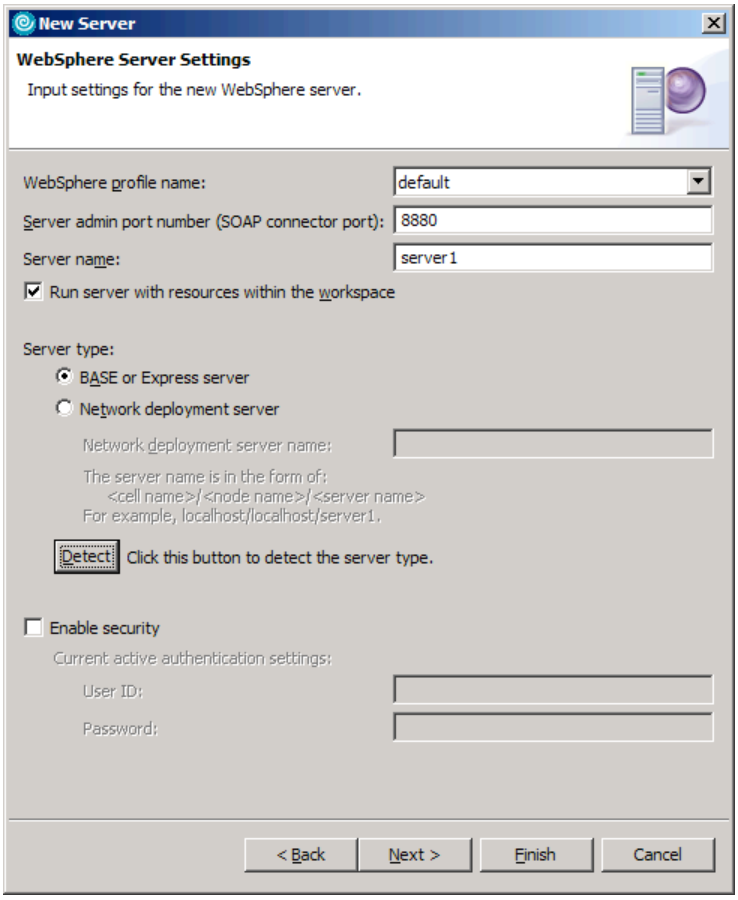
1. Seleccione la vista Servers y haga right-click en algún espacio vacío. Seleccione la opción New > Server del menú contextual.



2. En la primera pantalla del asistente “New Server”, escriba “localhost” como el Host Name y seleccione “Websphere 6.0 Server” como el server type tal y como se muestra en la siguiente ilustración. Pulse el botón “Next >” para continuar.



3. Finalmente, en la pantalla “WebSphere Server Settings”, seleccione las opciones por defecto, tal y como se muestra en la pantalla. Si el servidor está corriendo, puede presionar “Detect” para verificar estos ajustes. Presione el botón de “Finish” para terminar.


New Server
WebSphere Server Settings
Input settings for the new WebSphere server.
WebSphere profile name:
default
Server admin port number (SOAP connector port):
8880
Server name:
server1
Run server with resources within the workspace
Server type:
BASE or Express server
Network deployment server
Network deployment server name:
The server name is in the form of:
<cell name>/<node name>/<server name>
For example, localhost/localhost/server1.
Detect
Click this button to detect the server type.
Enable security
Current active authentication settings:
User ID:
Password:
< Back
Next >
Finish
Cancel