SOR

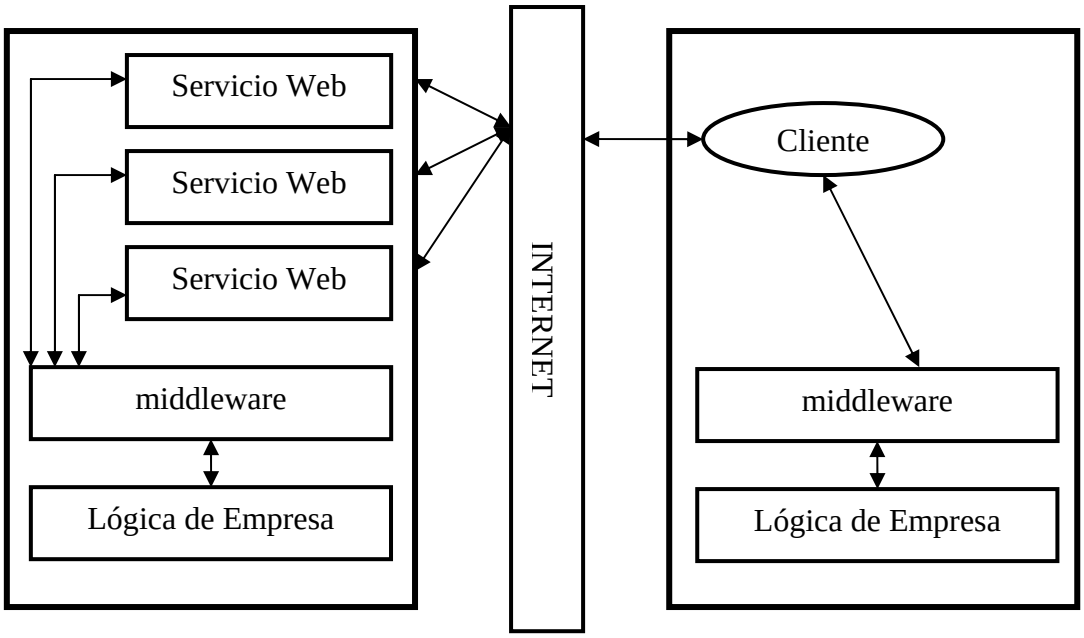
Introducción a Servicios Web

Se trata de un mecanismo relativamente nuevo que nos permite llamadas remotas a métodos a través de HTTP mediante el uso de protocolos y lenguajes basados en XML.
Los Servicios Web nos ofrecen una sencilla forma de extender la funcionalidad de nuestras aplicaciones empresariales. Con los Servicios Web podemos extender y permitir a otras aplicaciones acceder a nuestra lógica de empresa independientemente del lenguaje en el que esté escrita o la plataforma en la que corra.
El acceso es prácticamente como cuando solicitamos una página Web con la principal diferencia que a los Servicios Web accederemos a través de otra aplicación (Modelo de negocio B2B). El objetivo de los Servicios Web, podemos decir, que es la completa cooperación entre las aplicaciones existentes a través de la red de redes. Los Servicios Web resuelven las carencias de los modelos anteriores en las aplicaciones B2B.
Podemos prestar atención a las definiciones "oficiales" que existen sobre los Servicios Web.

Definición del World Wide Web Consortium (W3C): Una aplicación software identificada por un URI, cuyas interfaces se pueden definir, describir y descubrir mediante documentos XML. Un servicio Web soporta interacciones directas con otros agentes software utilizando mensajes XML intercambiados mediante protocolos basados en Internet.

Definición de la Webopedia: Una forma estándar de integrar aplicaciones basadas en Web utilizando los estándares abiertos XML, SOAP, WSDL y UDDI. XML se utiliza para etiquetar los datos, SOAP para transferir los datos, WSDL para describir los servicios disponibles y UDDI para listar que servicios están disponibles.

Los Servicios Web se han diseñado como fachada a la lógica de empresa de las distintas organizaciones.



Con ellos podemos acceder de una forma sencilla, como si de acceso local se tratase, a la lógica de empresa de otras organizaciones o departamentos de una misma organización.

Tecnología en Servicios Web

Los Servicios Web llevan asociados una serie de funcionalidades cada una de ellas relacionadas con una de las siguientes tecnologías que describimos a continuación.

SOAP: Es un protocolo de comunicación, basado en XML, para el intercambio de información en un entorno distribuido. Básicamente permite enviar y recibir respuestas en XML. Funciona sobre HTTP con las ventajas que nos aporta (estándar, transparencia de firewalls) pero también puede correr sobre otros protocolos conocidos (SMTP). Podemos decir que es independiente del protocolo de transporte subyacente. SOAP utiliza XML para definir la sintaxis de intercambio de mensajes de tal manera que cuando el cliente emita un mensaje SOAP para acceder a un método del Servicio Web el servidor recibe el mensaje, lo analiza y llama al método indicado en el mensaje SOAP enviado por el cliente, el servidor genera la respuesta SOAP con el resultado y se envía al cliente. El conocer SOAP es prácticamente como conocer HTTP, es decir, a la hora de la implementación será transparente para nosotros puesto que existen las APIS que nos facilitan dichas tareas.

WSDL: Se trata de un lenguaje basado en XML que se utiliza para describir todo lo que el cliente necesita saber para poder instanciar el Servicio Web. Nos permite describir en XML las operaciones y los tipos de datos soportados por el Servicio. Por cada Servicio Web tendremos un documento wsdl que nos describe el Servicio. Este documento está compuesto por tres partes

- La parte "Que" → En esta parte se define el mensaje y los tipos de datos intercambiados entre el cliente y el servidor. Definiríamos los métodos del Servicio, si el método es de entrada o salida (Parámetros que recibe y lo que devuelve).
- La parte "Como" → Describe los detalles de la implementación técnica de nuestro Servicio Web. Que protocolo de comunicación utilizaremos, si codificaremos la comunicación y enlazamos la parte del "Que" con el mecanismo de comunicación, el "Como".
- La parte "Donde" → Se refiere a la localización del Servicio Web. Se trata de la URI de localización del Servicio Web.

Si tenemos acceso al documento WSDL de un Servicio Web entonces será posible su uso.

UDDI: Si queremos tener acceso a los diferentes Servicios Web que se nos ofrece a través de la red o que nos pueden ofrecer otros departamentos de nuestra propia organización es obvio pensar que necesitamos una forma de localizar y registrar lo Servicios. De esta parte se encargará el protocolo UDDI. Conceptualmente podemos decir que UDDI es un registro o directorio el cual contiene información sobre los Servicios Web disponibles. Un servidor UDDI proporciona:

- Un conjunto de operaciones (vía SOAP) para registrar, eliminar y buscar Servicios Web.
- Existe una red pública de nodos donde en cada nodo se mantiene la misma información sobre los Servicios. Los nodos existentes se pueden encontrar en www.uddi.org. Aquí podemos registrar todos los datos referentes a nuestros Servicios Web. Este servidor UDDI manejará los siguientes datos:
  - Páginas blancas: Mantiene información general de la empresa
  - Páginas amarillas: Permite registrar el tipo de negocio facilitando la localización del Servicio requerido.
  - Páginas Verdes: Contiene información técnica sobre el Servicio Web.

Los nodos existentes son:

- http://uddi.ibm.com
- http://uddi.microsoft.com
- http://uddi.sap.com
- http://www.ntt.com/uddi

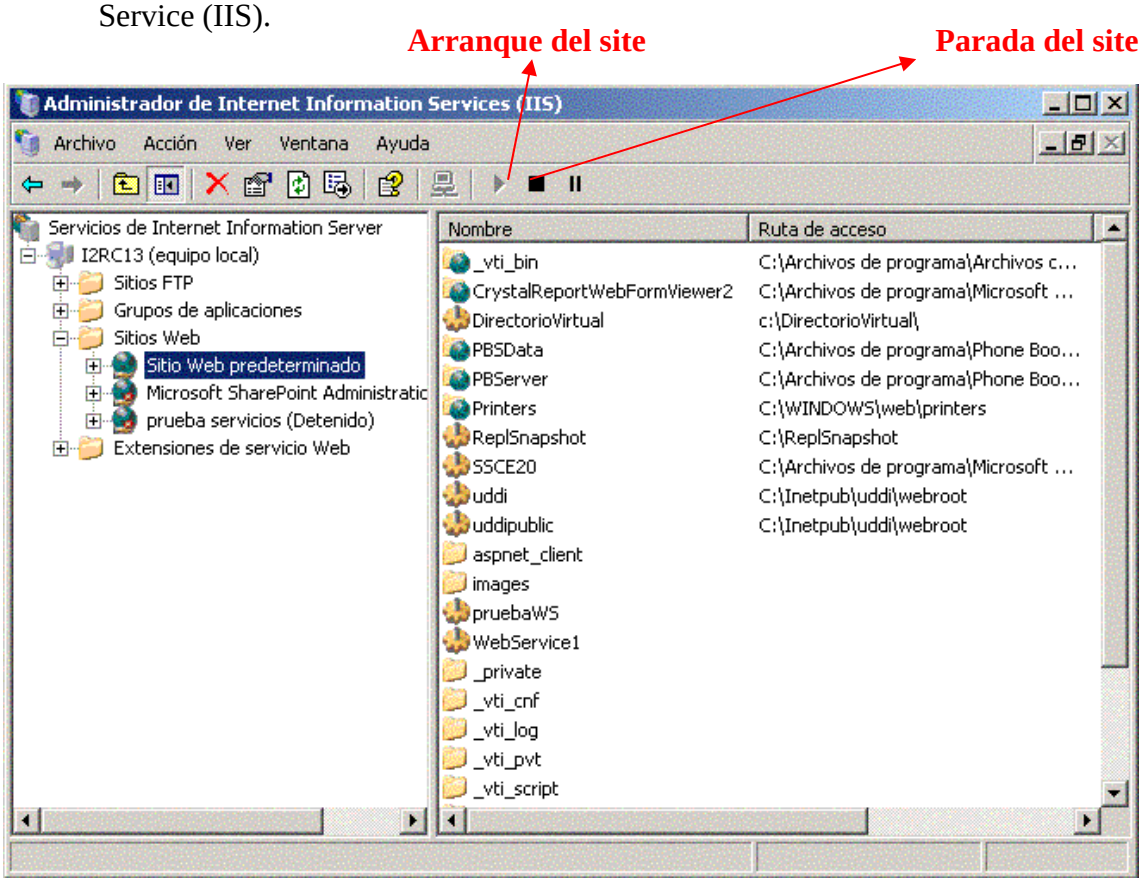
Podemos encontrar más información en http://www.uddi.org/register.html.
Microsoft además nos ofrece un servidor UDDI para Intranets en caso de que todos nuestros Servicios solo estén disponibles para nuestra Intranet. Su uso es exactamente igual que si lo hiciéramos en http://uddi.microsoft.com.

En los últimos tiempos se han ido eliminando estos servidores UDDI públicos.

Introducción al entorno de desarrollo Visual Studio .NET

En las práctica vamos a realizar los Servicios Web sobre Visual Studio .NET de Microsoft. Para la realización de los mismos las tecnologías que hemos visto antes van a ser prácticamente transparentes puesto que la propia plataforma genera todo lo necesario.
Comenzaremos explicando que necesitamos para poder realizar, registrar, publicar y consumir los Servicios Web en .NET.

- IIS → Es el Servidor Web que contendrá a los Servicios Web. Podemos acceder al administrador del IIS en Inicio → Panel de Control → Herramientas administrativas → Administrador de Internet Information Service (IIS).



El path del Sitio Web predeterminado si no lo hemos modificado será

C:\inetpub\wwwroot

Podemos crear un directorio dentro de este donde se encuentre nuestra aplicación y lo referenciaremos como

http://nombre_maquina/nombre_dir_aplicación/servicio.asmx

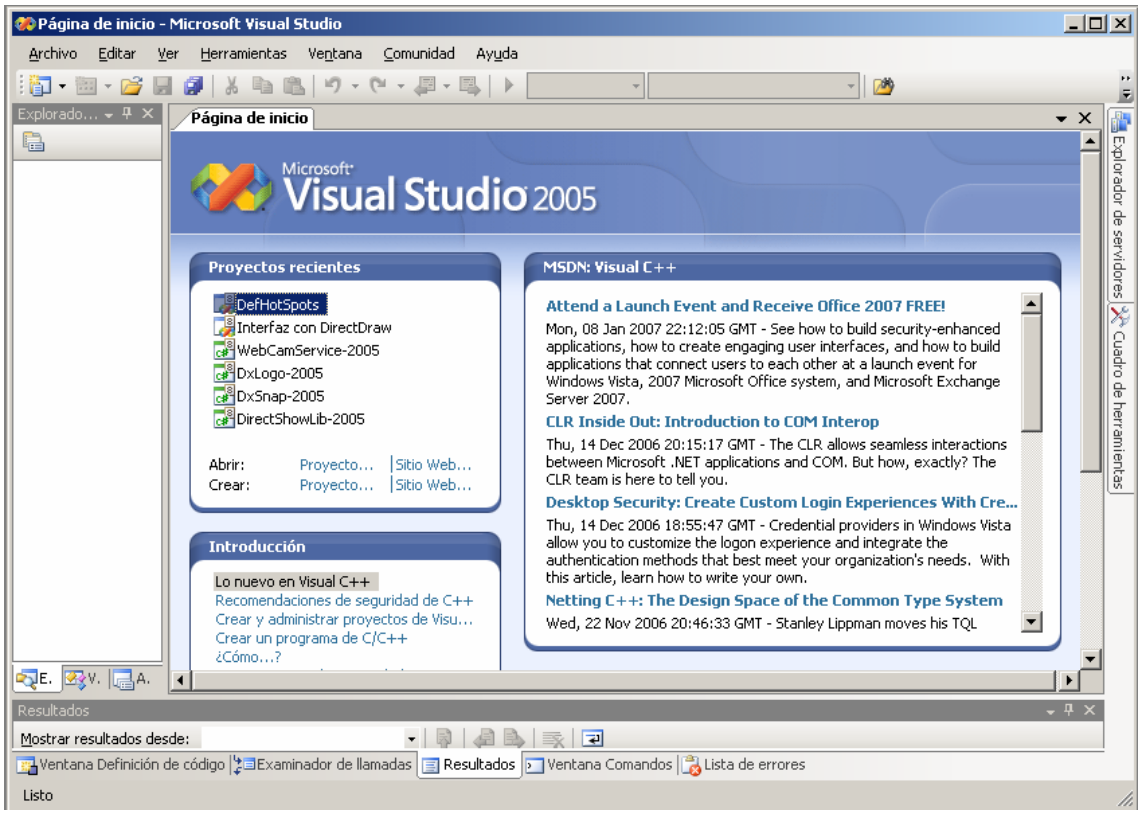
- Visual Studio .NET → Es el entorno de desarrollo de Microsoft .NET.

- UDDI para VS.NET → Es un servicio para el registro, publicación y la búsqueda de Servicios Web para entornos de Intranet. Viene con la versión de Windows 2003. Se instalará como un componente de Windows.
- SQL Server o MSDE → Gestor de base de datos necesario para que pueda funcionar el UDDI Service.

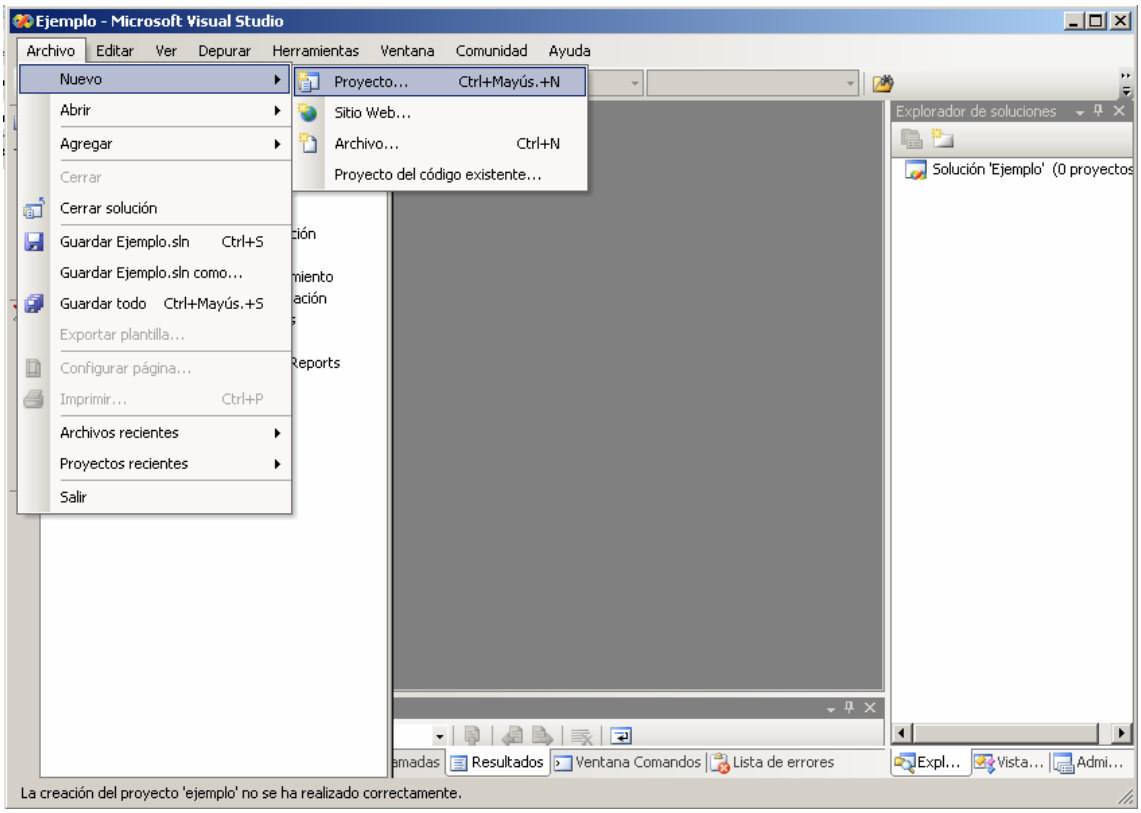
En el apartado de la publicación veremos como publicar un servicio en Internet y como es igual que publicarlo en el Directorio UDDI de la Intranet.

Creación de un Servicio Web

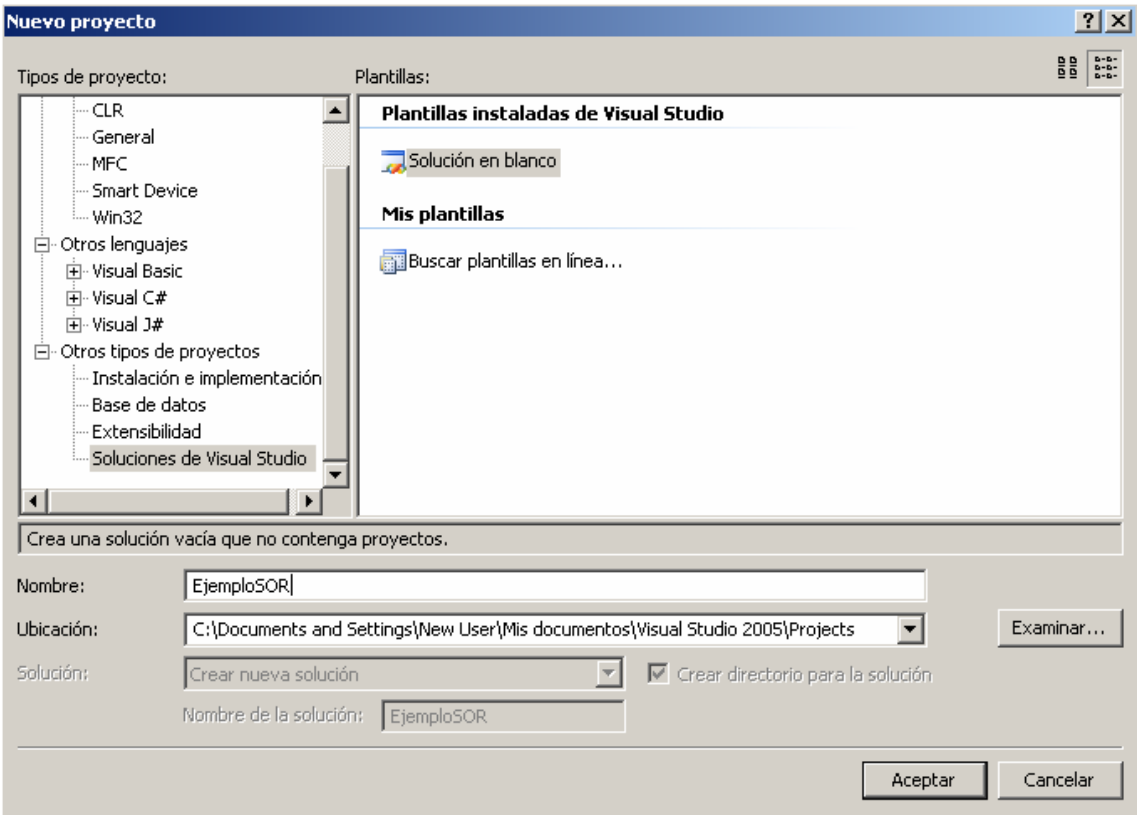
Primero debemos abrir el Visual Studio .NET.



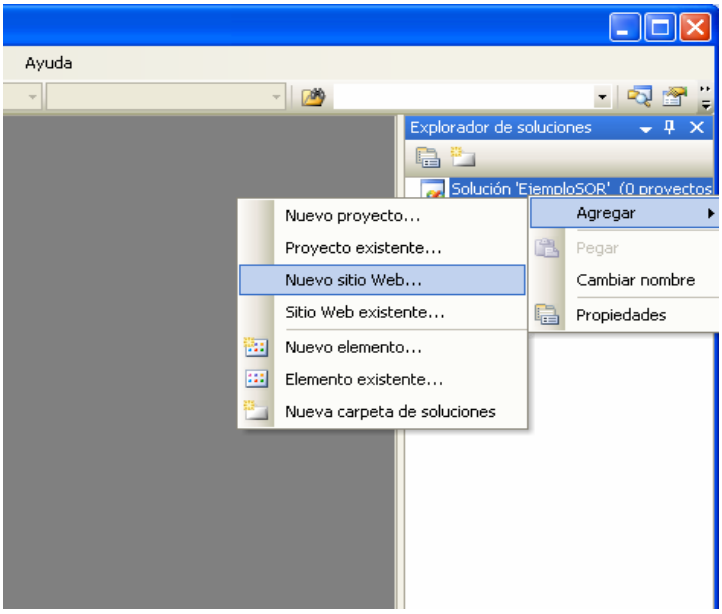
Creamos una nueva solución, que no es más que la agrupación lógica de un conjunto de proyectos.



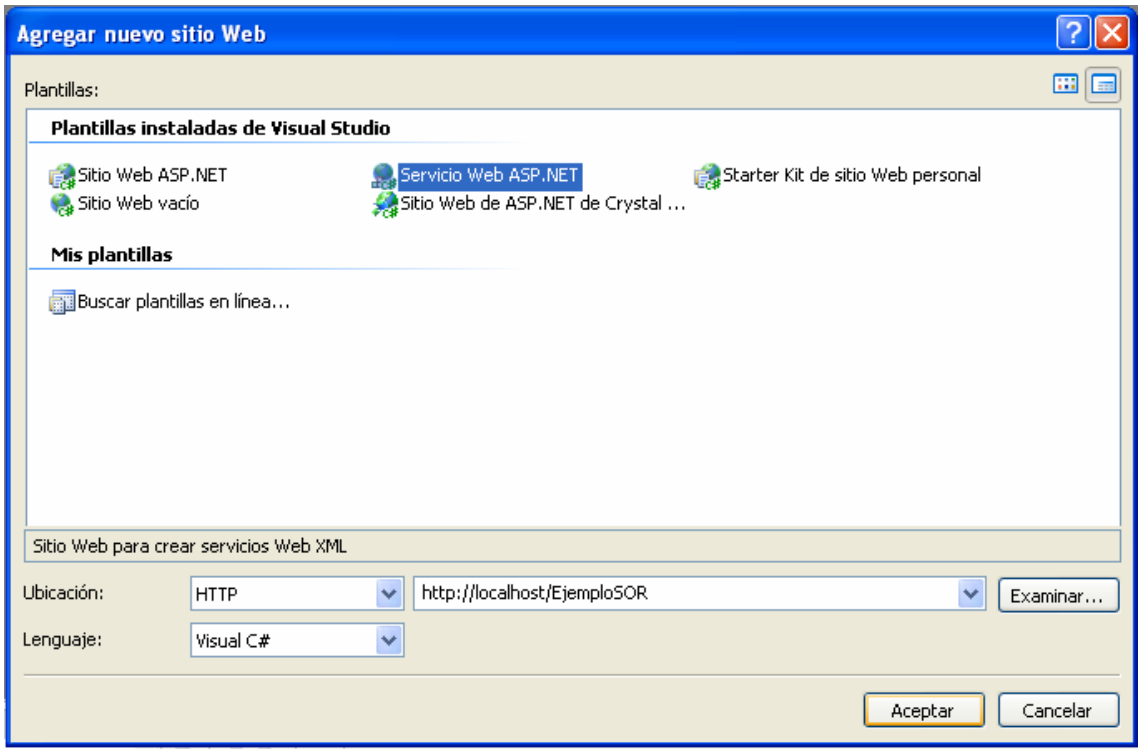
Al pulsar aparecerá una ventana en la cual podemos seleccionar una solución en blanco.



Cuando se cree la solución, situando el cursor encima y seleccionándola con el botón derecho podemos crear un sitio Web nuevo para comenzar a implementar el Servicio Web.

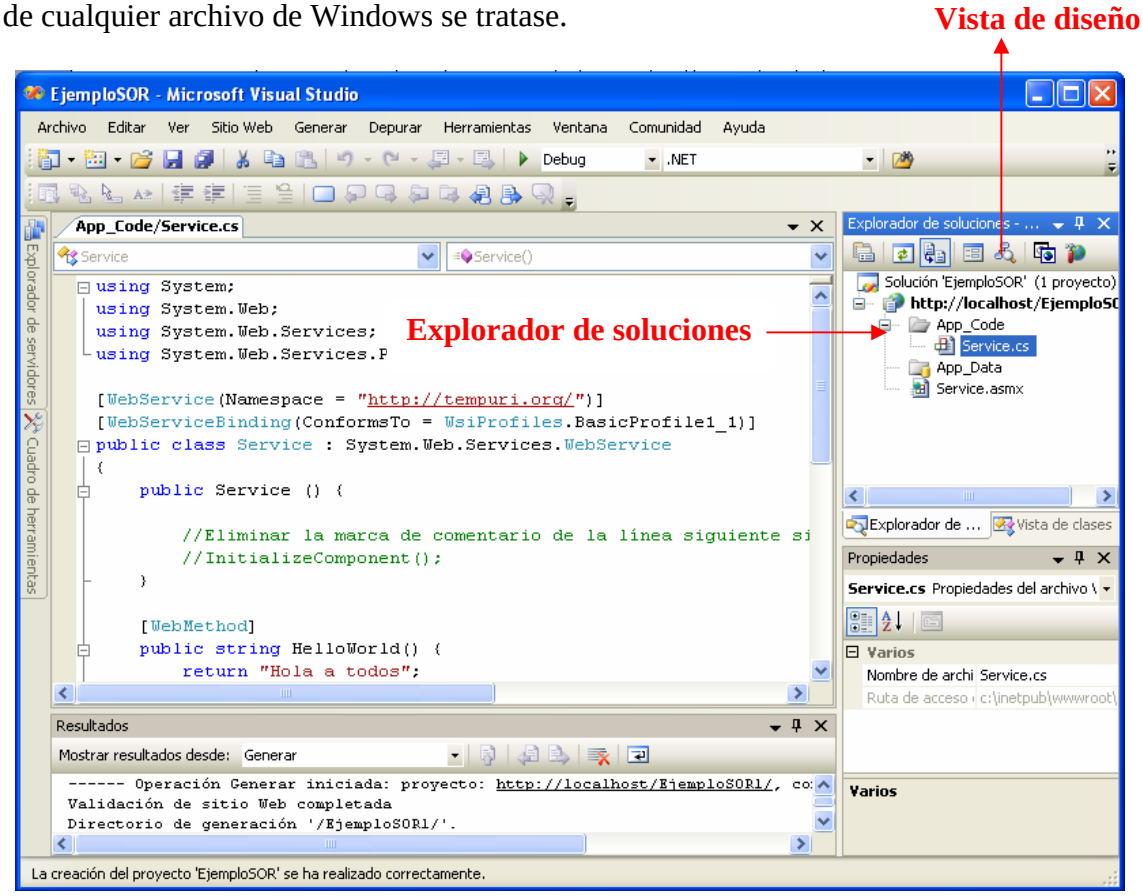


Una vez seleccionado aparece una ventana que nos permite indicar el tipo de aplicación Web que deseamos crear (en nuestro caso un Servicio Web). Además, en esta fase le podemos indicar el lenguaje en que queremos trabajar y la ubicación relativa a una URL o al sistema de archivos.

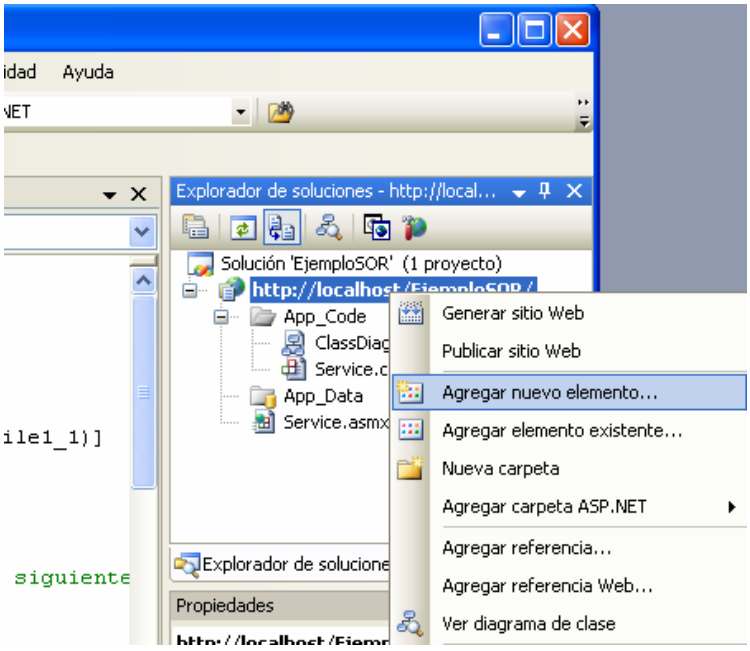


En este ejemplo como lenguaje hemos elegido C#, seleccionamos la plantilla Servicio Web ASP.NET y lo ubicaremos en el Sitio Web predeterminado (localhost), creando en él un directorio con el nombre EjemploSOR.

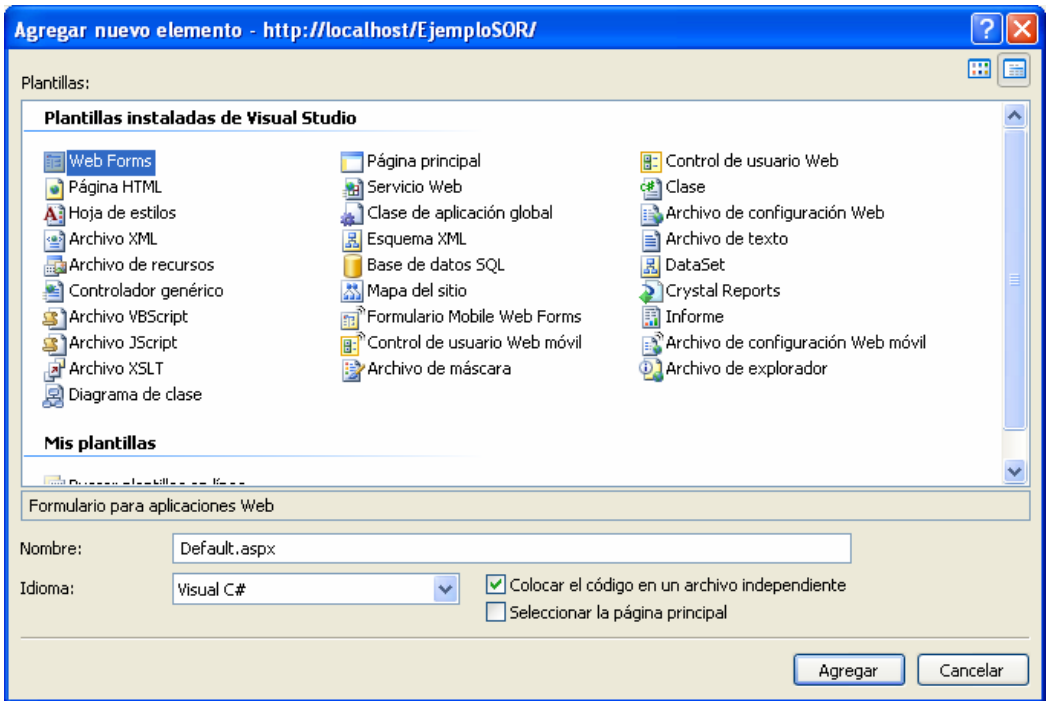
A continuación en el explorador de soluciones podemos ver como nos ha creado todos los archivos necesarios del proyecto junto con un servicio llamado Service1.asmx que le podemos cambiar el nombre posicionándonos sobre él y pinchando dos veces como si de cualquier archivo de Windows se tratase.



Podemos crear nuevos proyectos dentro de la solución siguiendo el procedimiento realizado para crear el Servicio Web. Si en lugar de un nuevo proyecto queremos añadir elementos al proyecto creado se realizará de la siguiente forma.



Seguidamente aparecerá una ventana con los elementos que podemos añadir (Servicios Web, clases, etc.).



El código del servicio Web creado podemos modificarlo para continuar con el ejemplo.

```
using System;
using System.Web;
using System.Collections;
using System.Web.Services;
using System.Web.Services.Protocols;


/// <summary>
/// Descripción breve de WebService
/// </summary>
[WebService(Namespace = "EjemploSOR")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class WebService : System.Web.Services.WebService {

    public WebService () {

        //Eliminar la marca de comentario de la línea siguiente si
utiliza los componentes diseñados
        //InitializeComponent();
    }

    [WebMethod]
    public string HelloWorld() {
        return "Hola a todos";
    }
}
```

Primero modificaremos el namespace (modo de agruparlo) para que no se pueda confundir con otros Servicios Web de otros proveedores. Se trata de una forma única de

identificar a un Servicio Web, que en este caso podría coincidir con el nombre de otro Servicio.

```
[WebService(Namespace = "EjemploSOR")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
```

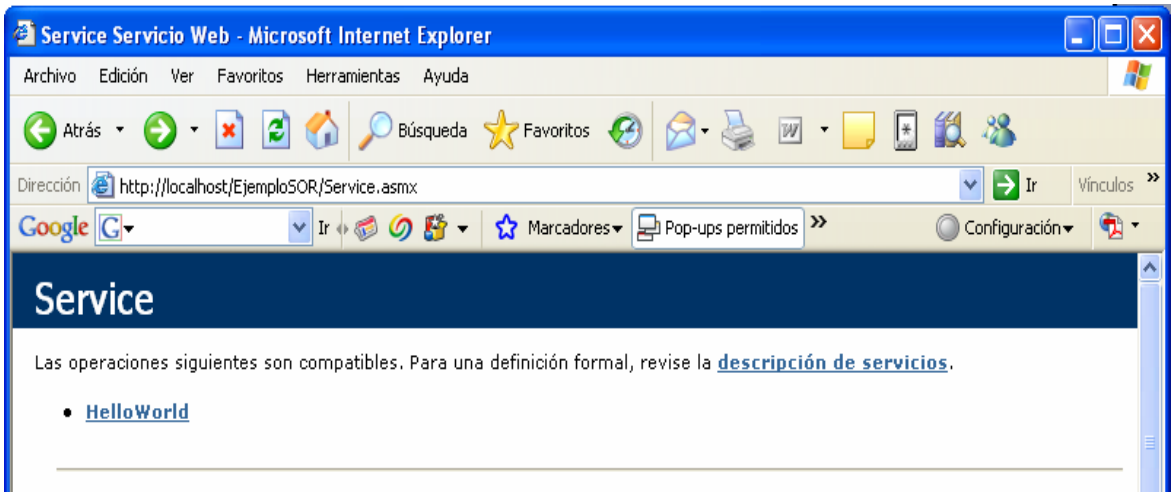
Vemos como el propio VS.NET nos crea una función comentada llamada HelloWorld. Encima aparece la etiqueta [WebMethod] la cual nos sirve para exponer un método como parte de un Servicio Web. Cualquier método que lleve esta etiqueta deberá ser de tipo público. Con este atributo, también podemos indicar una descripción, la cual será expuesta por el servicio Web, de esta forma los usuarios tendrán una mejor idea de cual es la función de dicho método.

```
WebMethod]
public string HelloWorld() {
    return "Hola a todos";
}
```

A continuación, se quitan los comentarios de la parte del código que afecta a la función HolaMundo y generaremos el proyecto (F5). Lo generará en el Servidor Web que le hemos indicado al crear el proyecto y podemos probar el servicio accediendo en el navegador con la URL

http://servidor_web/EjemploSOR/nombre_servicio.asmx

Un Servicio Web no se realiza para probar en un navegador si no que son diseñados para ser consumidos por otra aplicación usando protocolos tales como HTTP GET/POST, SMTP o SOAP sobre HTTP. SOAP es el más apropiado para la comunicación servidor a servidor. Como hemos visto .NET nos provee de un mecanismo para poder probar el servicio como si de una página Web se tratase (se trata de una plantilla hecha en ASP.NET la cual se encarga de generar la página para acceder el Servicio Web desde el navegador).



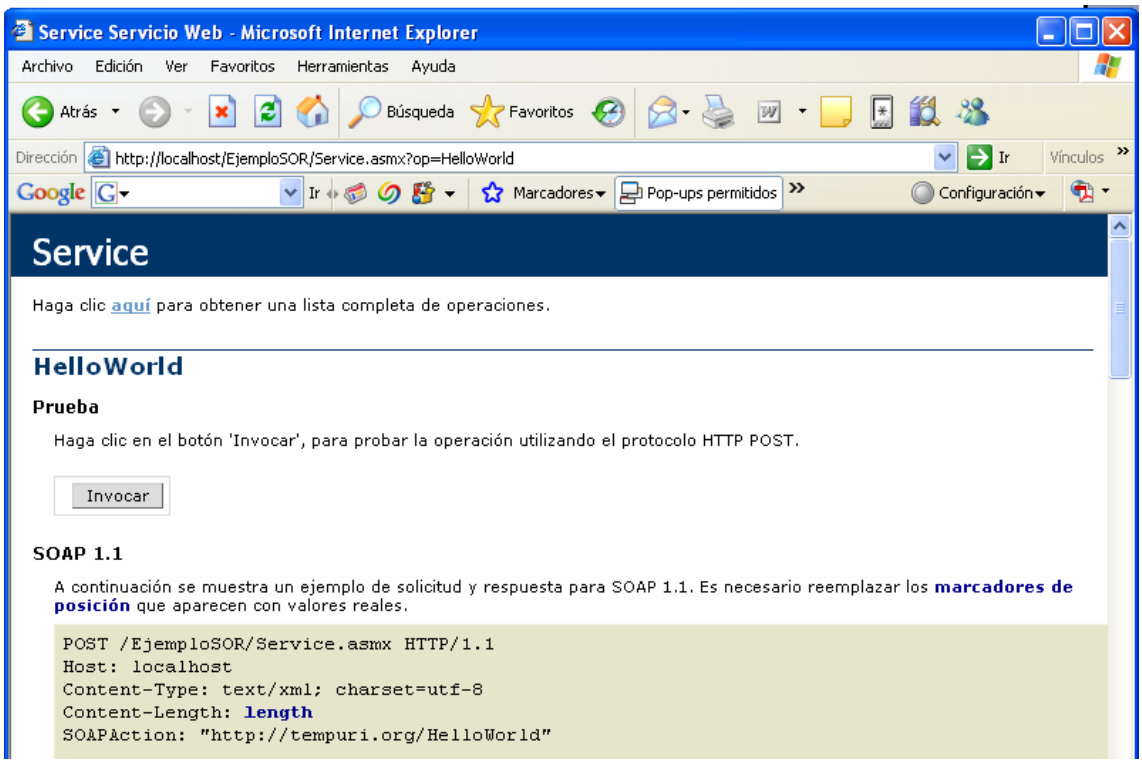
Pinchando en descripción de servicios podemos ver el documento WSDL que VS.NET ha generado para el Servicio en concreto.

```
<?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:tns="http://tempuri.org/" xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    targetNamespace="http://tempuri.org/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
- <wsdl:types>
- <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
- <s:element name="HelloWorld">
  <s:complexType />
    </s:element>
- <s:element name="HelloWorldResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="HelloWorldResult" type="s:string" />
    </s:sequence>
    </s:complexType>
    </s:element>
    </s:schema>
    </wsdl:types>
- <wsdl:message name="HelloWorldSoapIn">
  <wsdl:part name="parameters" element="tns:HelloWorld" />
    </wsdl:message>
- <wsdl:message name="HelloWorldSoapOut">
  <wsdl:part name="parameters" element="tns:HelloWorldResponse" />
    </wsdl:message>
- <wsdl:portType name="ServiceSoap">
- <wsdl:operation name="HelloWorld">
  <wsdl:input message="tns:HelloWorldSoapIn" />
  <wsdl:output message="tns:HelloWorldSoapOut" />
    </wsdl:operation>
    </wsdl:portType>
- <wsdl:binding name="ServiceSoap" type="tns:ServiceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="HelloWorld">
  <soap:operation soapAction="http://tempuri.org/HelloWorld" style="document" />
- <wsdl:input>
  <soap:body use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
    </wsdl:binding>
- <wsdl:binding name="ServiceSoap12" type="tns:ServiceSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="HelloWorld">
  <soap12:operation soapAction="http://tempuri.org/HelloWorld" style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
    </wsdl:binding>
```

```
- <wsdl:service name="Service">
- <wsdl:port name="ServiceSoap" binding="tns:ServiceSoap">
  <soap:address location="http://localhost/EjemploSOR/Service.asmx" />
    </wsdl:port>
- <wsdl:port name="ServiceSoap12" binding="tns:ServiceSoap12">
  <soap12:address location="http://localhost/EjemploSOR/Service.asmx" />
   </wsdl:port>
   </wsdl:service>
   </wsdl:definitions>
```

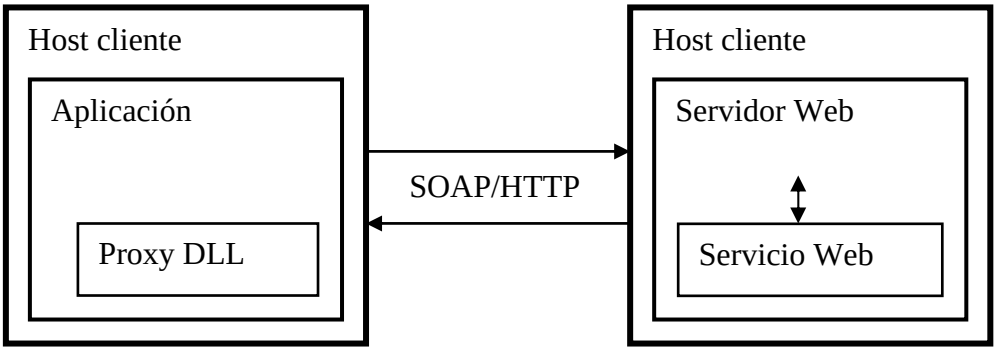
Este documento nos describirá los métodos soportados, los tipos de entrada y salida de cada método. El protocolo de comunicación y la localización del Servicio. Esta información es muy importante para la aplicación que quiere consumir el Servicio Web.

Si pinchamos en el link que pone HelloWorld nos lleva a otra pantalla donde podemos invocar al Servicio Web por medio de HTTP POST (nos devolverá un XML como respuesta pero no una respuesta con formato SOAP si no usando HTTP POST) y además nos describe un ejemplo de solicitud y respuesta con SOAP.



Consumo de Servicios Web

Una vez creado un Servicio Web veremos como invocarlo desde una aplicación cliente. Todavía no vamos a entrar en el registro de servicios si no que de momento supondremos que conocemos lo necesario del Servicio Web como para poder consumirlo. Existen varias formas de acceso a un Servicio Web. Nosotros nos centraremos en la clase proxy que implementa las llamadas mediante SOAP pero de forma transparente a nosotros. Esta clase proxy se puede generar de dos maneras: Con la utilidad en línea de comando WSDL.exe o desde el propio VS.NET.



WSDL.exe

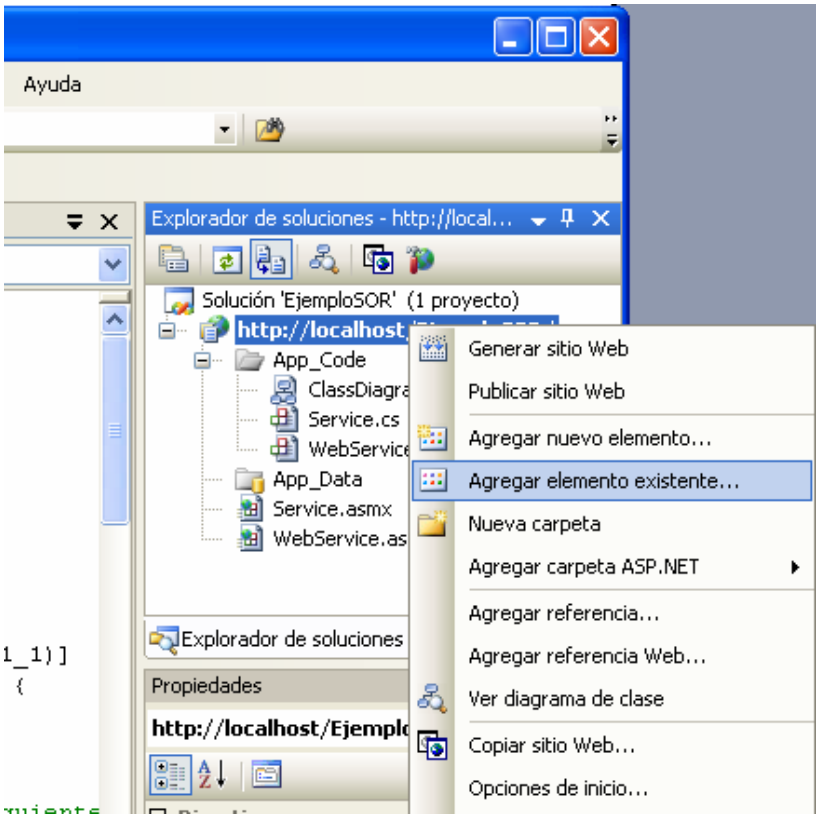
Su utilización produce un fichero del lenguaje que estemos utilizando (.cs si es C#, .vb si es VB.NET…) que implementa la clase proxy a partir de la descripción WSDL del Servicio.

wsdl http://nombre_servidor/EjemploSOR/nombre_servicio.asmx?WSDL

La ruta del comando será

…..\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin

Una vez se ha generado la clase proxy añadiremos dicha clase al proyecto

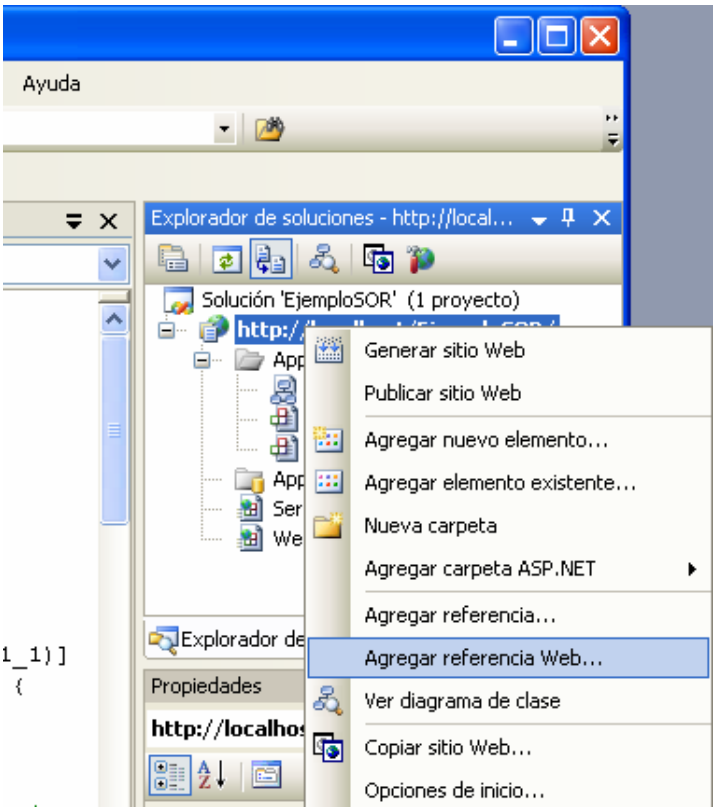


Por defecto la clase se colocará en el espacio de nombres raiz por lo que no hará falta añadir el espacio de nombres con la directiva *using*, no obstante se puede indicar el espacio de nombres donde queremos situarla con la opción de *WSDL.exe /n:<namespace>*.

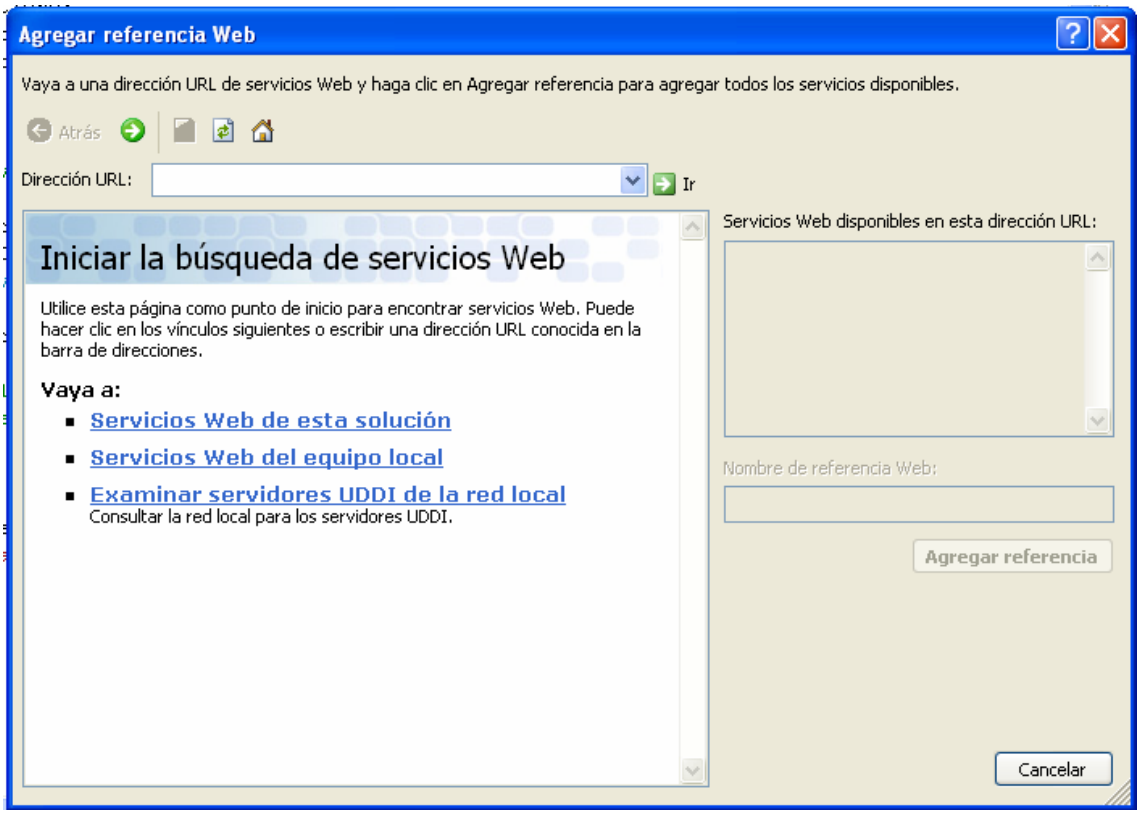
Esta clase proxy ya se encarga de llamar a las funciones del Servicio Web haciendolo transparente para nosotros que simplemente actuaremos con la clase proxy como una clase local.

VS.NET

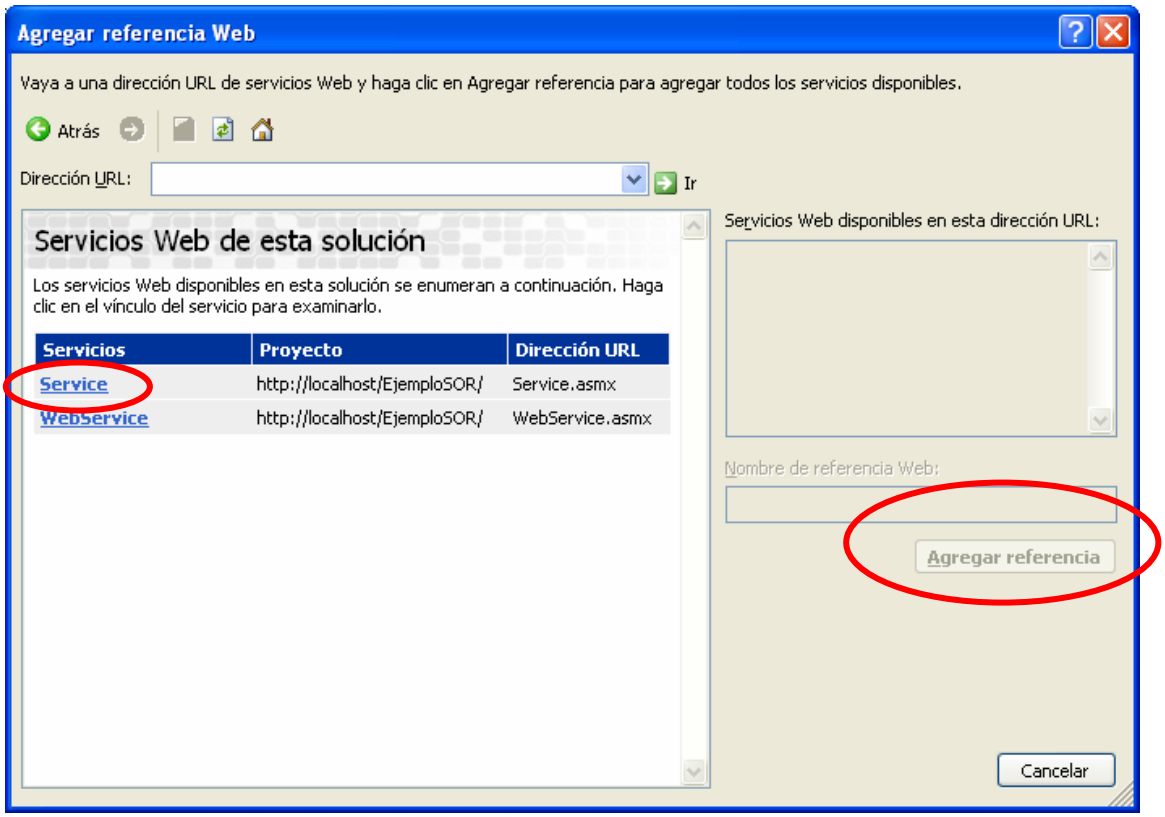
Una forma más cómoda de crear una clase proxy para consumir un Servicio Web es el uso del entorno de desarrollo.
Para ello nos situaremos sobre el proyecto desde el que queremos invocar al Servicio Web y añadiremos una referencia al Servicio Web.



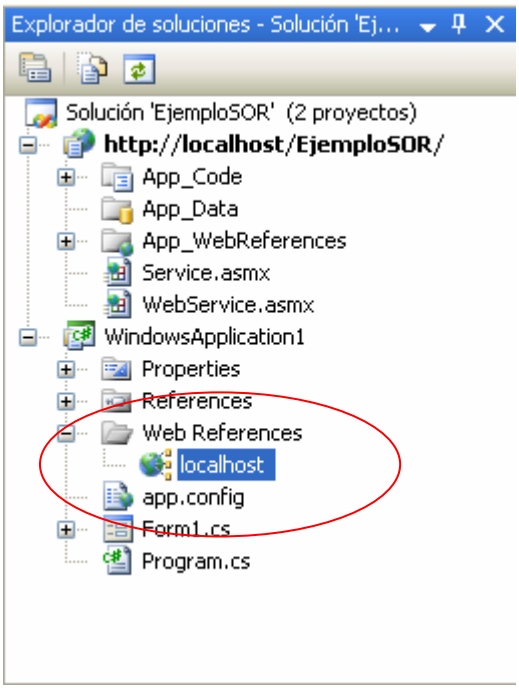
En el momento que pinchemos obtendremos una ventana que nos indica varias localizaciones para buscar el Servicio Web deseado.



En este caso para probarlo y puesto que no hemos realizado la publicación en ningún Servidor UDDI pincharemos sobre *Servicios Web del equipo local* y obtendremos los Servicios Web que hemos creado.



Pincharemos sobre el Servicio Web deseado y pulsaremos en el botón *Agregar referencia*.
Con estos pasos hemos conseguido agregar la referencia al Servicio Web en nuestro proyecto.



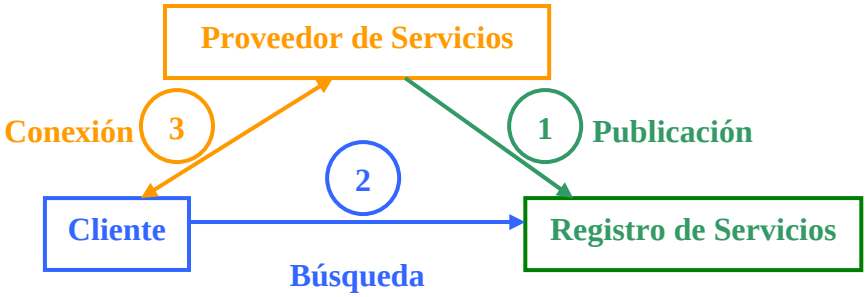
Podemos cambiar el nombre de la referencia *localhost* por el que queramos. Después simplemente en nuestra clase cliente (por ejemplo una página ASP.NET) debemos añadir la referencia con la directiva *using EjemploSOR_cliente.localhost* y ya podemos instanciar el Servicio como si de una clase local se tratase o bien escribiendo el namespace (localhost) seguido del nombre del servicio.

```
private void Button1_Click(object sender, System.EventArgs e)
{
  localhost.Service sr = new localhost.Service();
  TextBox1.Text = sr.HelloWorld();
}
```

En el ejemplo estamos invocando el Servicio Web y llamando a su función HelloWrold la cual nos devolverá una cadena de texto que introduciremos en el TextBox de la aplicación cliente.

Publicación y descubrimiento de Servicios Web

El modelo de los Servicios Web es muy sencillo. Tenemos un proveedor que se encarga de ofrecer su lógica de negocio a través de un Servicio Web. Este proveedor debe registrar el Servicio Web en algún servidor de registro para que el cliente pueda localizarlo y por tanto consumirlo. El cliente lo encontrará en el registro de Servicios y accederá a lo consumirá en la localización que aparezca en el registro de Servicios.



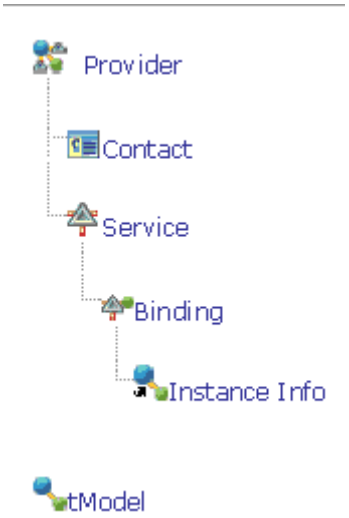
Por ahora hemos visto como un proveedor genera el Servicio Web y como un cliente lo consume pero no lo hemos utilizado un Servicio de registro de Servicios Web (UDDI). Como comentamos antes existe un Registro de Servicios Web a nivel de Internet con una serie de nodos de empresas importantes del sector, donde se encuentra la información replicada. Nosotros en la práctica vamos a hacer uso del Servicio UDDI que proporciona Microsoft para registro, búsqueda y publicación de Servicios en Intranets. Este es exactamente igual que el que ofrece MS en su nodo de Internet (las mismas funcionalidades y el mismo aspecto visual.)

Registro de Servicios

Debido a que actualmente están quietando los registros públicos y que el registro UDDI de Intranet no se encuentra disponible en los sistemas instalados en los laboratorios, esta parte no se hará en prácticas. Se explica de manera informativa.

El Servicio UDDI tiene 6 entidades que debemos comprender primeramente.

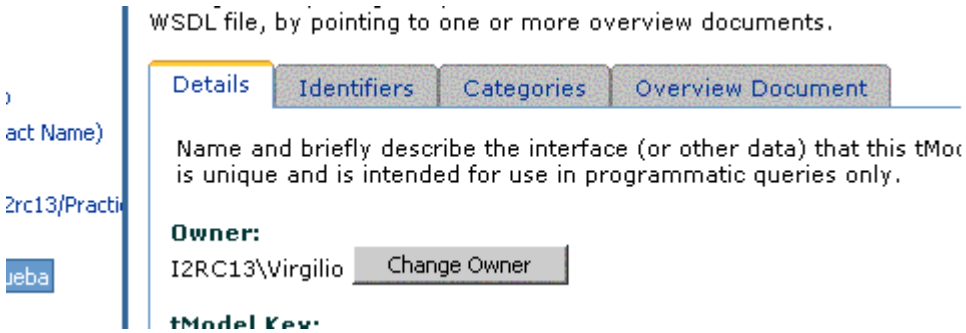
- Provider: Persona, grupo u organización que ofrece una serie de Servicios Web XML.
- Contact: Persona de contacto para informarte sobre los temas relacionados con los Servicios del proveedor correspondiente.
- Service: EL Servicio Web
- Binding: El punto de acceso al Servicio Web o URL del Servicio.
- Instace info: Referencia a un tModel que contiene información técnica relevante sobre un Binding.
- tModel: Información técnica sobre el interfaz, como el WSDL, también puede representar una unidad organizativa con datos descriptivos, como un identificador.



Para registrar el Servicio abriremos un navegador y escribiremos la siguiente URL:

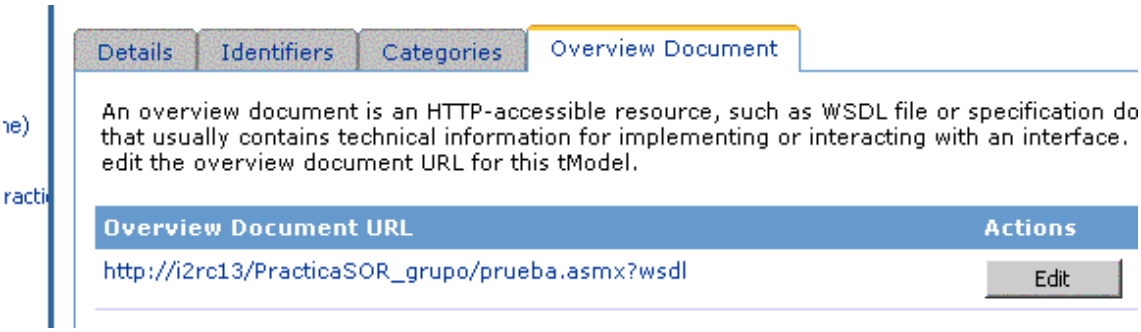
http://nombre_servidor/uddi

Una vez me muestre la página pulsaré la opción de Publish. Pincharé en la pestaña tModel para crear la información de los Servicios Web (sus documentos WSDL).
En los tModels definiremos la ruta al los documentos WSDL del Servicio Web.



Definiremos los detalles del tModel y posteriormente la referencia al documento WSDL en Overview Document. En nuestro caso sería:

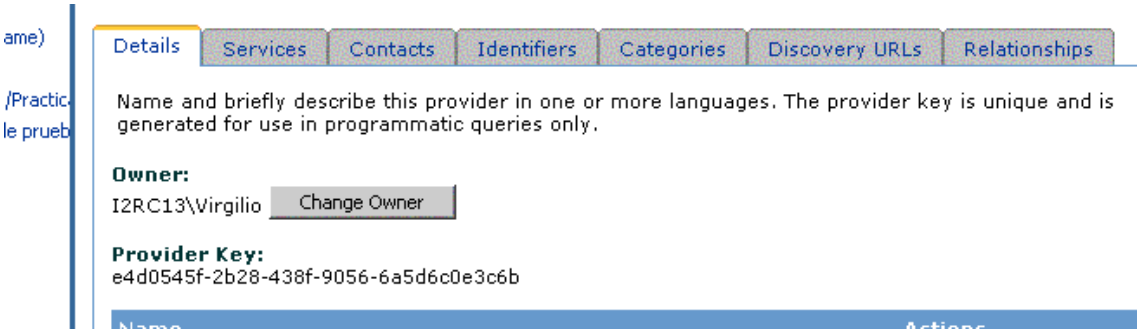
http://nombre_servidor/EjemploSOR/Service.asmx?wsdl



Las otras dos pestañas se utilizarían para clasificar de una manera más exacta el tModel.

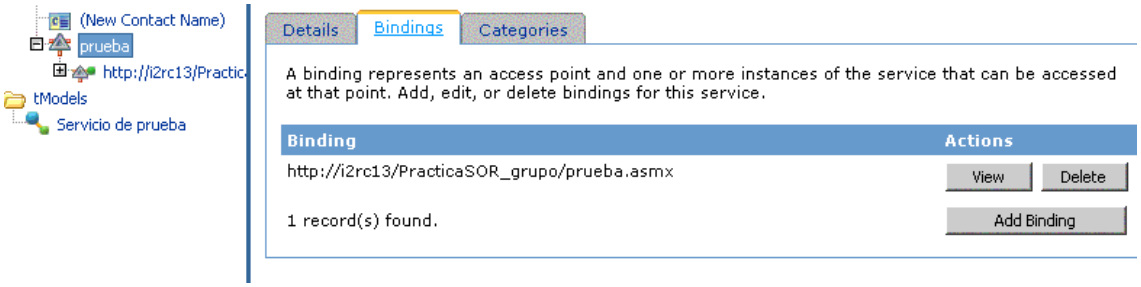
Una vez listo el tModel pasaremos a registrar el proveedor y relacionarlo con el tModel.

Pincharé en la pestaña *Provideer* y añadiré un nuevo proveedor y su descripción correspondiente. Añadiremos todos los datos necesarios de contacto para que un cliente nos pueda localizar en un momento dado.

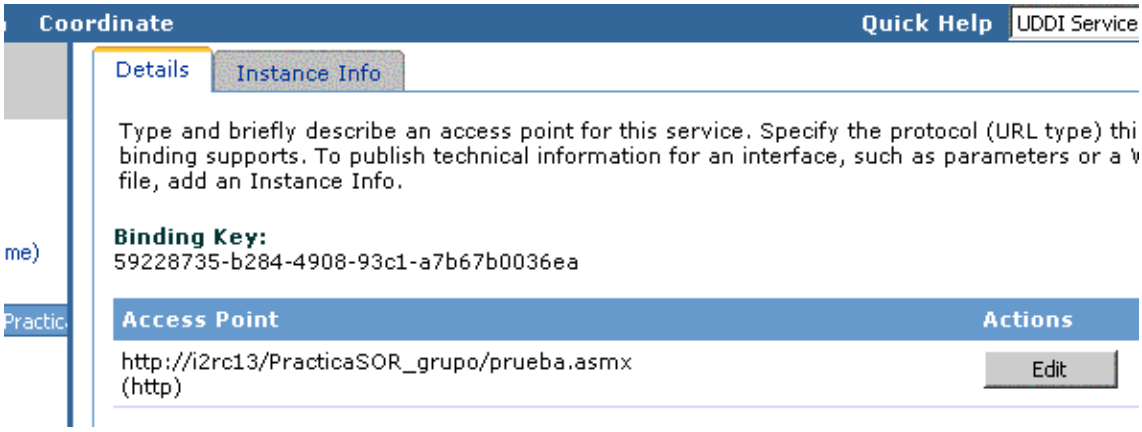


Posteriormente añadiremos un Servicio para este proveedor pinchando en la pestaña *Service*. Para el Servicio añadiremos los detalles y el Binding. El Binding para nuestro Servicio Web sería

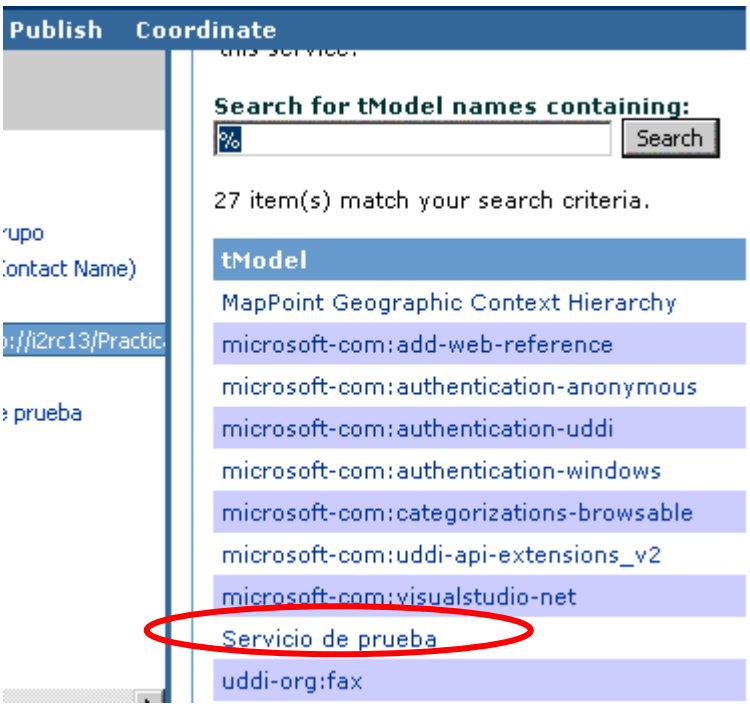
http://nombre_servidor/EjemploSOR/Service.asmx



Cuando editamos o visualizamos el Binding creado se introducen los detalles y el Instante Info que no es más que la asociación con un tModel creado anteriormente.



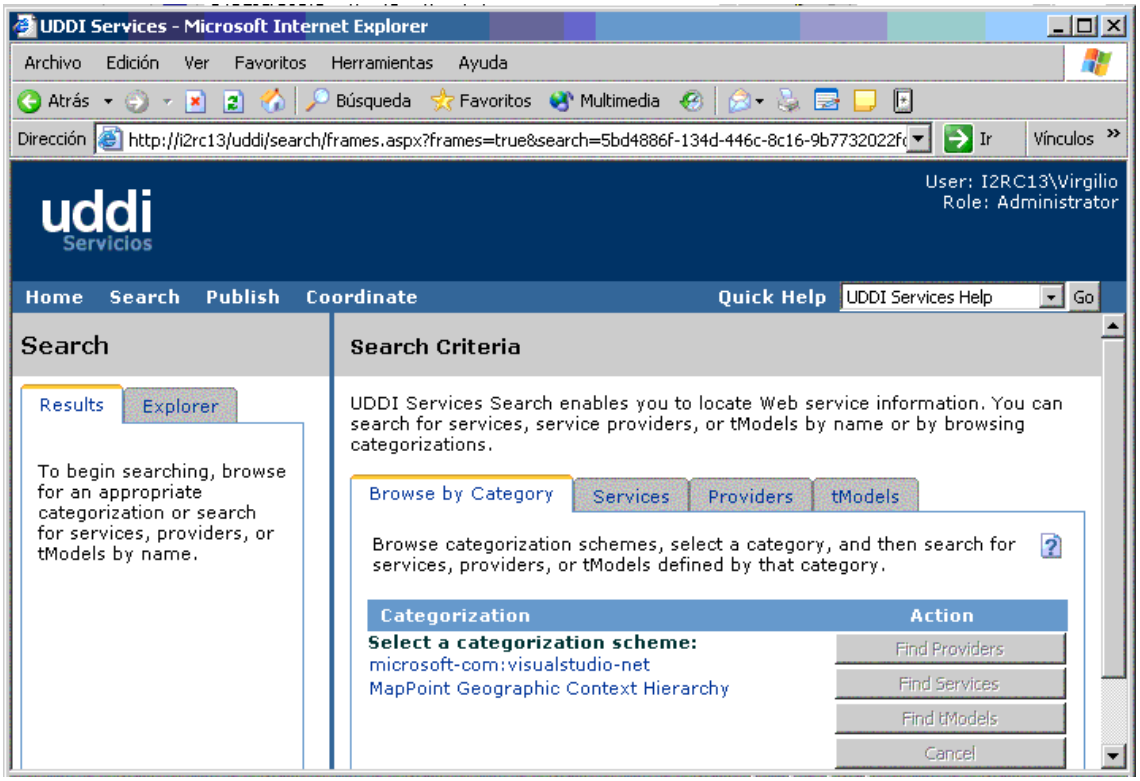
En esta parte indicamos el punto de acceso que habíamos comentado y mediante el instante info (pulsando en añadir instante info) podemos buscar el tModel correspondiente entre los existentes.



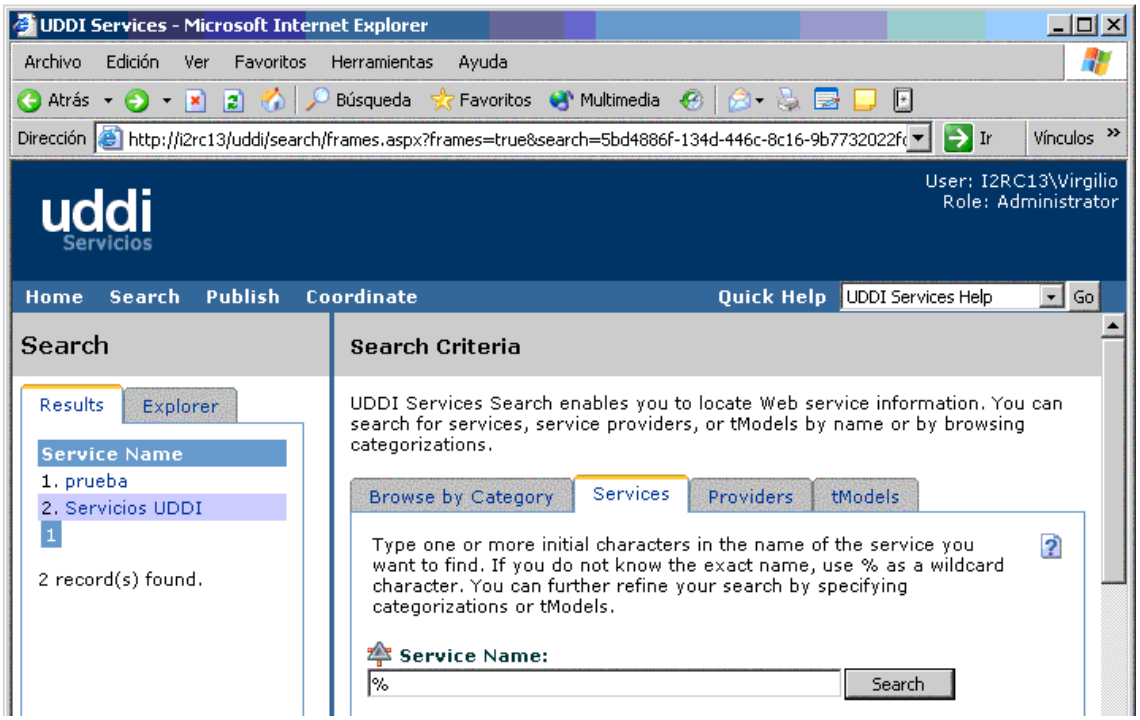
Pinchando en el tModel correspondiente relacionamos el Servicio que queremos registrar con la información técnica que hemos registrado anteriormente. De esta manera desde un Servicio podremos acceder al documento WSDL para poder consumir un Servicio Web.

Búsqueda de Servicios

Para finalizar una vez registrado el Servicio cualquier otro grupo de prácticas podrá localizar nuestros Servicios así como contactar con nosotros para obtener más información.
Accediendo a la misma URL que en el caso anterior pulsaremos en la opción de búsqueda (Search).



Podemos realizar la búsqueda por el criterio que más nos interese. Por la categoría en la que se engloba el Servicio, por los proveedores, por los propios Servicios o por los tModels. En este caso lo haremos por los Servcios.



Hemos indicado que nos muestre todos los servicios existentes y en el lado izquierdo podemos ver como se nos muestra dicha lista. Aquí elegiremos el Servicio que deseemos consumir y obtendremos su información técnica para poder instanciarlo (WSDL).
Cuando tengamos el acceso al documento WSDL podemos añadir la referencia y acceder a él.

Referencias

Programming .NET WebService, O'Reilly, Alex Ferrara & Matthew MacDonald
http://www.microsoft.com/spanish/msdn/articulos/archivo/120402/voices/vbtchgettingstartedwithxmlwebservicesinvisualstudionet.asp
http://www.programacion.com/tutorial/xmlrpcsoap/
http://www.microsoft.com/spanish/msdn/articulos/archivo/281201/voices/service10032001.asp
http://www.microsoft.com/spanish/msdn/articulos/archivo/281201/voices/service10172001.asp
http://greco.dit.upm.es/~tomas/cursos/isi/trabajos/2002/saguirre_t.pdf