Desarrollador de Aplicaciones Web
Programación Web III

Departamento de Ingeniería e Investigaciones Tecnológicas

Servicios Web

Ing. Mariano Juiz
Ing. Matias Paz Wasiuchnik
Ing. Pablo Nicolás Sanchez

Agenda

- Introducción
- Estándares
- Infraestructura
- Arquitecturas basadas en Servicios Web
- Implementación en .Net

Introducción

- Internet proporciona contenido a los usuarios, de diversas maneras.
- Las páginas web devuelven contenido HTML, que se presenta al usuario mediante una interfaz (browser).
- Con la evolución de las páginas web en aplicaciones web, estas pueden “comunicarse” mediante servicios evitando la intervención del usuario.
- Los dispositivos móviles pueden consumir contenido de Internet.
- Las aplicaciones web deben “ajustarse” a los dispositivos móviles o, mejor aún, ofrecerle servicios.

Introducción

Introducción

Historia

Se ha intentado crear estándares sin éxito:

- DCOM (Microsoft)
  - Sistema Operativo: Microsoft
  - Lenguajes de Programación: Visual Basic, C++, C.
- CORBA (múltiples vendedores)
  - Independiente del Sistema Operativo
  - Lenguajes de Programación: Java, C, C++, Visual Basic. etc-.
- RMI (Invocación Remota de Métodos)
  - Independiente del Sistema Operativo
  - Lenguaje de Programación: Java.

Dependientes de la implementación que realizaba el vendedor.

Al usar RPC (Remote Procedure Call), generaban problemas de seguridad o eran bloqueados por los Firewalls.

Introducción

RPC (Remote Procedure Calls)

- Es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos.
- Muy utilizados en la arquitectura Cliente-Servidor.
- Algunos lo llaman la primera generación de Servicios Web.
- Suele ser implementado por medio del mapeo de servicios directamente a funciones específicas del lenguaje
- Existen implementaciones diversas (Microsoft, Sun, OSF, etc.)

Introducción

XML-RPC (Llamadas a Procedimientos Remotos)

- Fue el primer mecanismo que surgió para invocar procedimientos remotos vía XML, ofrece una manera muy sencilla de invocar operaciones en sistemas heterogéneos a través de una estructura simple.
- Es una implementación menos robusta para llevar acabo una intercomunicación en XML que **SOAP**.

Algunas implementaciones conocidas son:

- **Apache XML-RPC**,[5] una implementación en Java.
- **XMLRPC-EPI**,[6] una implementación C.
- **XML-RPC-C**,[7] una implementación para C y C++.

Introducción

Servicios Web

“Son sistemas software diseñados para soportar una interacción interoperable máquina a máquina sobre una red.
Los Servicios Web suelen ser APIs Web que pueden ser accedidas dentro de una red (principalmente Internet) y son ejecutados en el sistema que los ofrece“ - W3C -

Objetivo

Ofrecer **interoperabilidad** entre **plataformas**, **lenguajes de programación** y **sistemas operativos,** sin la intervención del usuario

Protocolos Utilizados

No fue pensado para un protocolo en particular, es decir, nada nos impide utilizar SOAP sobre algún otro protocolo de Internet (SMTP, FTP, etc.).

Se utiliza principalmente el protocolo HTTP por ser muy difundido

Estándares

XML (Extensible Markup Language)

El XML es una especificación desarrollada por **W3C**.

Lenguaje para describir documentos con estructura estándar (conocido en la industria)

Permite a los desarrolladores crear sus propios tags, que les permiten habilitar definiciones, transmisiones, validaciones, e interpretación de los datos entre aplicaciones y entre organizaciones.

XML Schemas – XSD

- Basado en XML
- Lenguaje para descibrir tipo de datos

Estándares

SOAP (Simple Object Access Protocol)

Es un protocolo que permite la comunicación entre aplicaciones a través de mensajes por medio de Internet.

Está basado en XML, es independiente de la plataforma, y del lenguaje.

Mensajes
Un mensaje SOAP es una simple documento XML que contiene los siguientes elementos:

- Un elemento de sobres que identifica el documento XML como un mensaje SOAP
- Un elemento de encabezado que contiene la información de cabecera
- Un elemento del cuerpo que contiene información de la llamada y la respuesta
- Un elemento de error que contiene errores y la información de estado

Estándares

SOAP (Simple Object Access Protocol) - Ejemplo

Request

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetails xmlns="http://warehouse.example.com/ws">
      <productId>827635</productId>
    </getProductDetails>
  </soap:Body>
</soap:Envelope>
```

Response

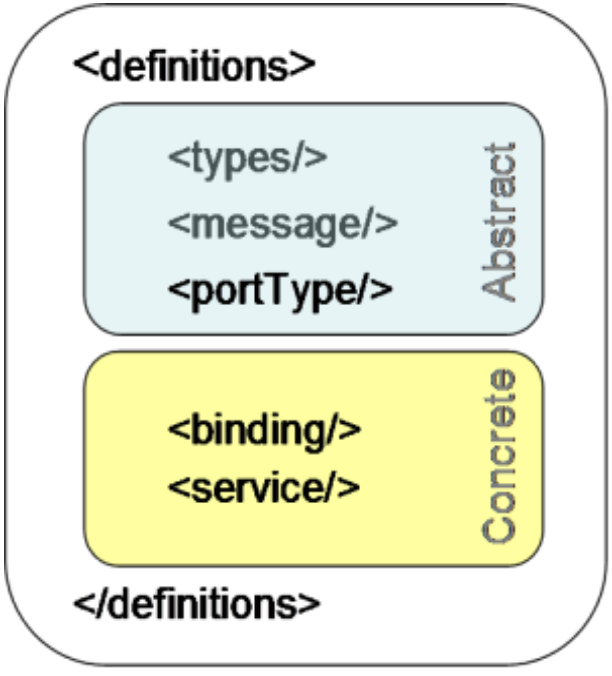
```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetailsResponse xmlns="http://warehouse.example.com/ws">
      <getProductDetailsResult>
        <productName>Toptimate 3-Piece Set</productName>
        <productId>827635</productId>
        <description>3-Piece luggage set.  Black Polyester.</description>
        <price>96.50</price>
        <inStock>true</inStock>
      </getProductDetailsResult>
    </getProductDetailsResponse>
  </soap:Body>
 </soap:Envelope>
```

Estándares

WSDL (Web Services Description Language)

Lenguaje basado en XML que indica las interfaces que provee el Servicio Web y los tipos de datos necesarios para la utilización del mismo. Actualmente en versión 2.0.

- Define tipo de datos (XSD)
- Define mensajes a partir de tipos
- Define detalles de protocolo de transporte y formato de SOAP
    - SOAP 1.1, HTTP Get/Post, MIME



Estándares

WSDL (Web Services Description Language)

Tipos de Datos

<types>: Esta sección define los tipos de datos usados en los mensajes. Se utilizan los tipos definidos en la especificación de esquemas XML.

Mensajes

<message>: Define los elementos de mensaje. Cada mensaje puede consistir en una serie de partes lógicas. Las partes pueden ser de cualquiera de los tipos definidos en la sección anterior.

Tipos de Puerto

<portType>: Define las operaciones permitidas y los mensajes intercambiados en el Servicio.

Bindings

<binding>: Especificamos los protocolos de comunicación usados.

Servicios

<service>: Conjunto de puertos y dirección de los mismos. Esta parte final hace referencia a lo aportado por las secciones anteriores. Con estos elementos no sabemos que hace un servicio pero si disponemos de la información necesaria para interactuar con él funciones, mensajes de entrada/salida, protocolos...).

Estándares

DISCO

Document Discovery

Permite que un cliente encuentre un servicio web en particular

Documento que especifica servicios descubiertos en una URL en particular

Utiliza formato XML

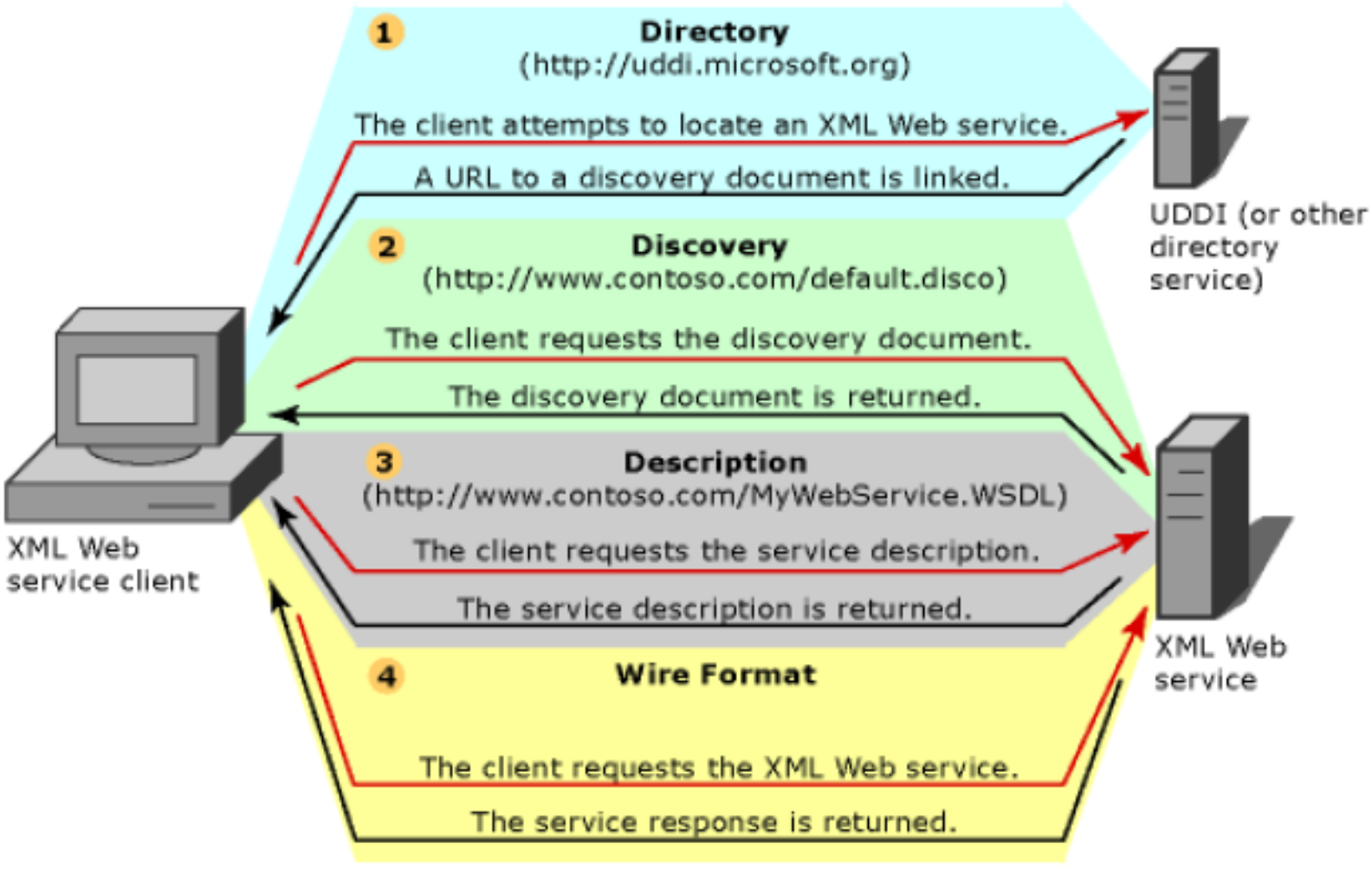
Los documentos tienen extensiones .disco and .vsdisco

UDDI (Universal Description, Discovery and Integration) – www.uddi.org

Es un servicio de directorio donde las empresas pueden registrar y buscar Servicios Web, conocer sus capacidades, ubicación, y requerimientos en un formato reconocido universalmente.

UDDI utiliza **WSDL** para describir las interfaces de los Servicios Web. Por ende, utiliza HTTP, XML y SOAP.

Infraestructura



Tecnologías utilizadas

Modo estándar de representar datos

XML (and XML schemas)

Formato de mensaje común y extensible

SOAP

Lenguaje de contrato común y extensible

Web Services Description Language (WSDL)

Modo estándar de descubrir servicios

Disco

Modo estándar de descubrir proveedores de servicio

Universal Description, Discovery, and Integration (UDDI)

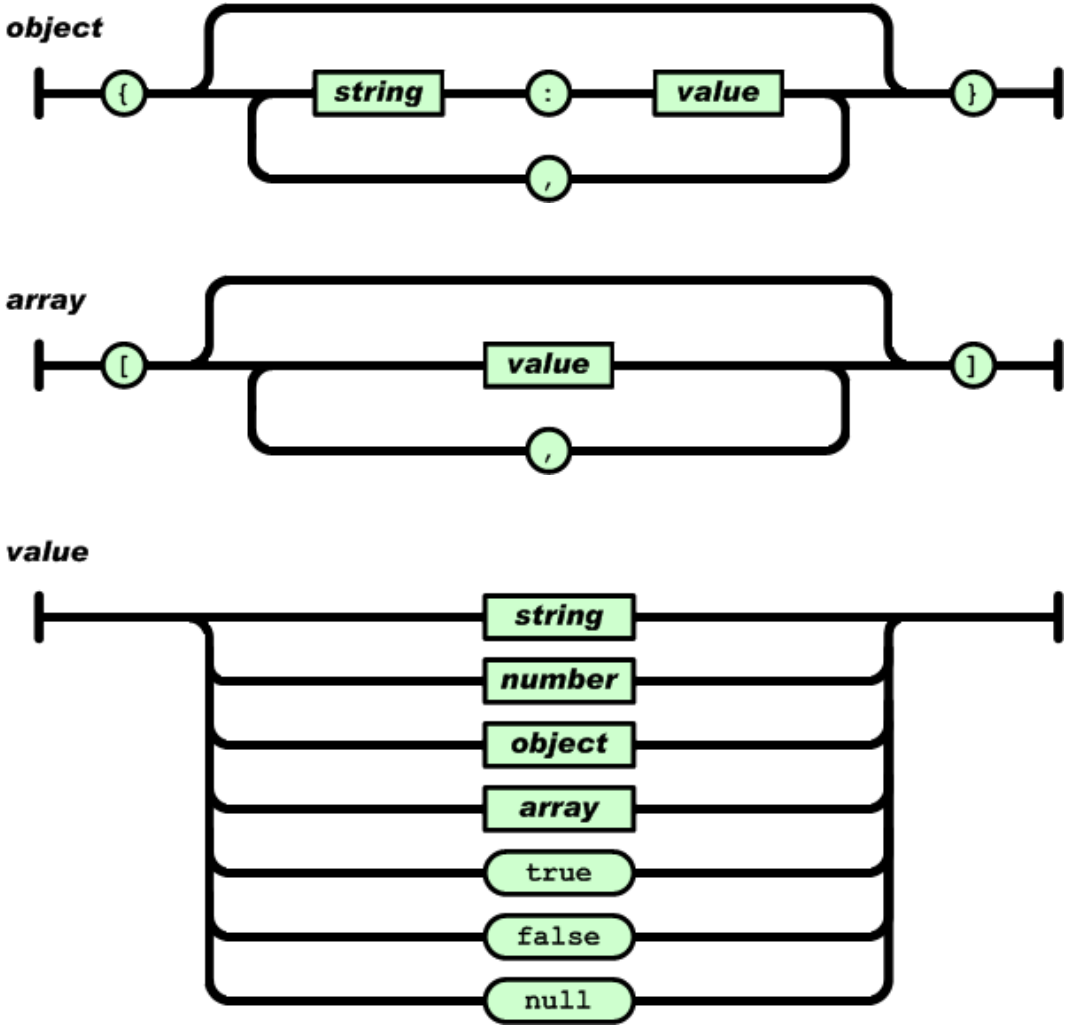
Otras Arquitecturas

REST (REpresentation **S**tate **T**ransfer**)**

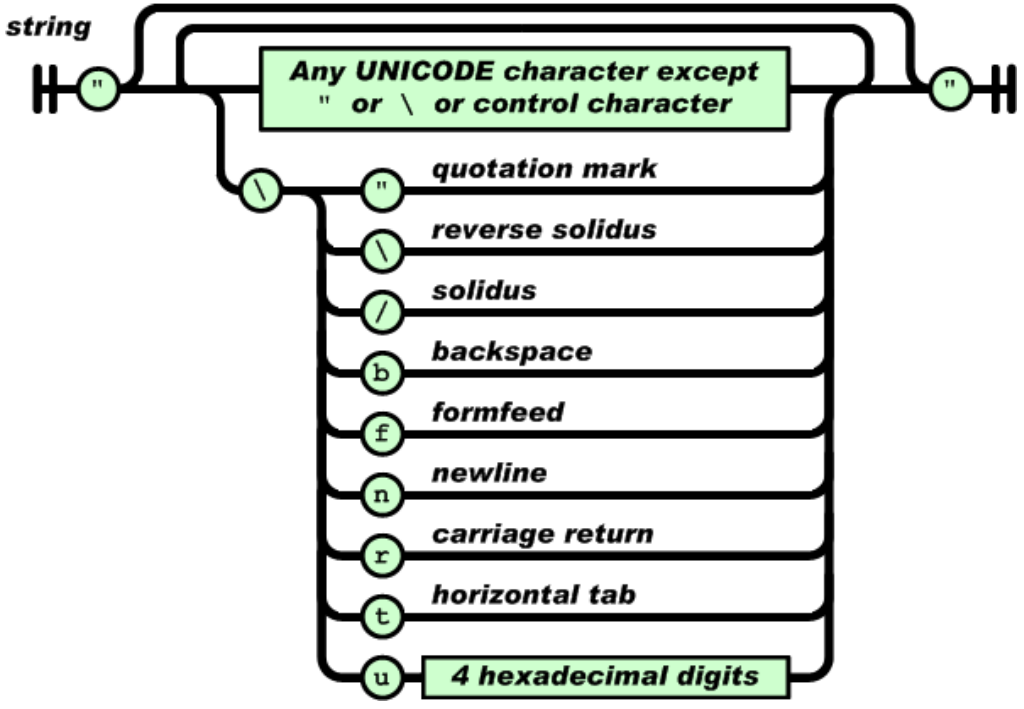
- Es un estilo de arquitectura que ofrece un buen desempeño, escalabilidad y desacoplamiento
- Puede implementarse con cualquier protocolo, pero su mayor implementación es en **HTTP**.
- Comunicación **sin estado** (enfoque fire and forget)
  - Cada petición HTTP contiene toda la información necesaria para responder a la petición, sin necesidad que el cliente ni el servidor tenga que recordar el estado de su comunicación.
  - Verbos HTTP
    - **GET** (copia de solo lectura)
    - **PUT** (cambiar)
    - **POST** (añadir)
    - **DELETE** (eliminar)
  - Se usa CACHE para optimizar
    - Todas las transiciones de estado son mediante el uso de links, provistos por el servidor
- Representación de los recursos
  - Al acceder al URI podemos obtener HTML, PDF, XML, etc.

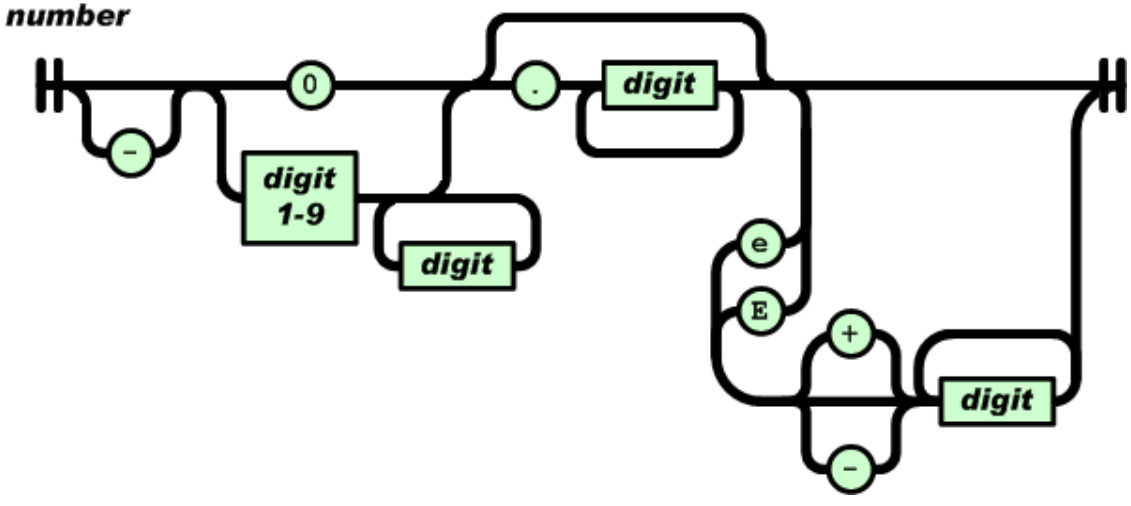
JSON (JavaScript Object Notation)

- Es un formato liviano de intercambio de datos
- Constituido por dos estructuras:



JSON (JavaScript Object Notation)





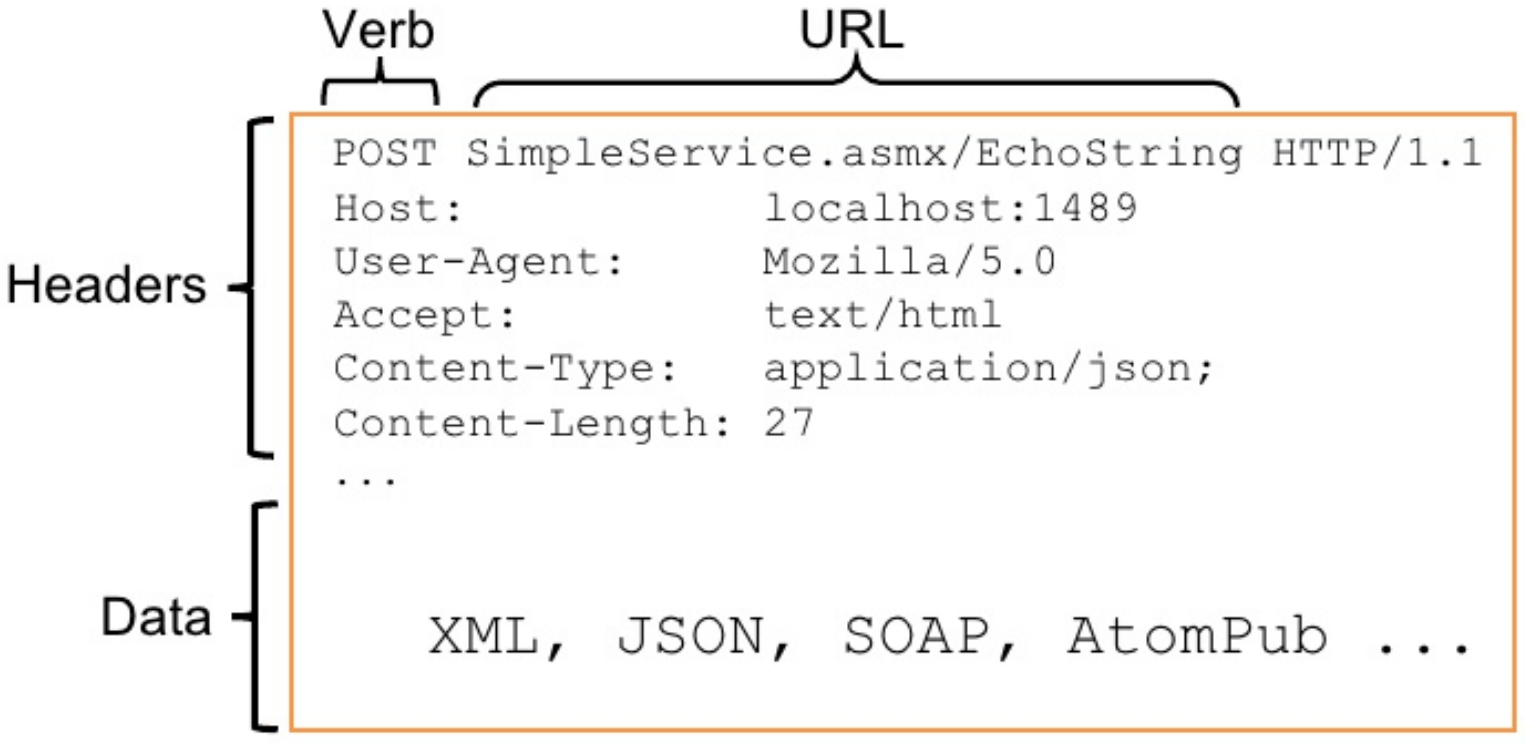
JSON (JavaScript Object Notation)

XML String	JSON String

<classinfo>
<students>
<student>
<name>Michael Smith</name>
<average>99.5</average>
<age>17</age>
<graduating>true</graduating>
</student>
<student>
<name>Steve Johnson</name>
<average>34.87</average>
<age>17</age>
<graduating>false</graduating>
</students>
</classinfo>
	{ "classinfo" :
	{
	"students" : [
	{
	"name" : "Michael Smith",
	"average" : 99.5,
	"age" : 17,
	"graduating" : true
	},
	{
	"name" : "Steve Johnson",
	"average" : 34.87,
	"age" : 17,
	"graduating" : false
	}]
	}
	}

Implementación ASMX

Comunicación HTTP



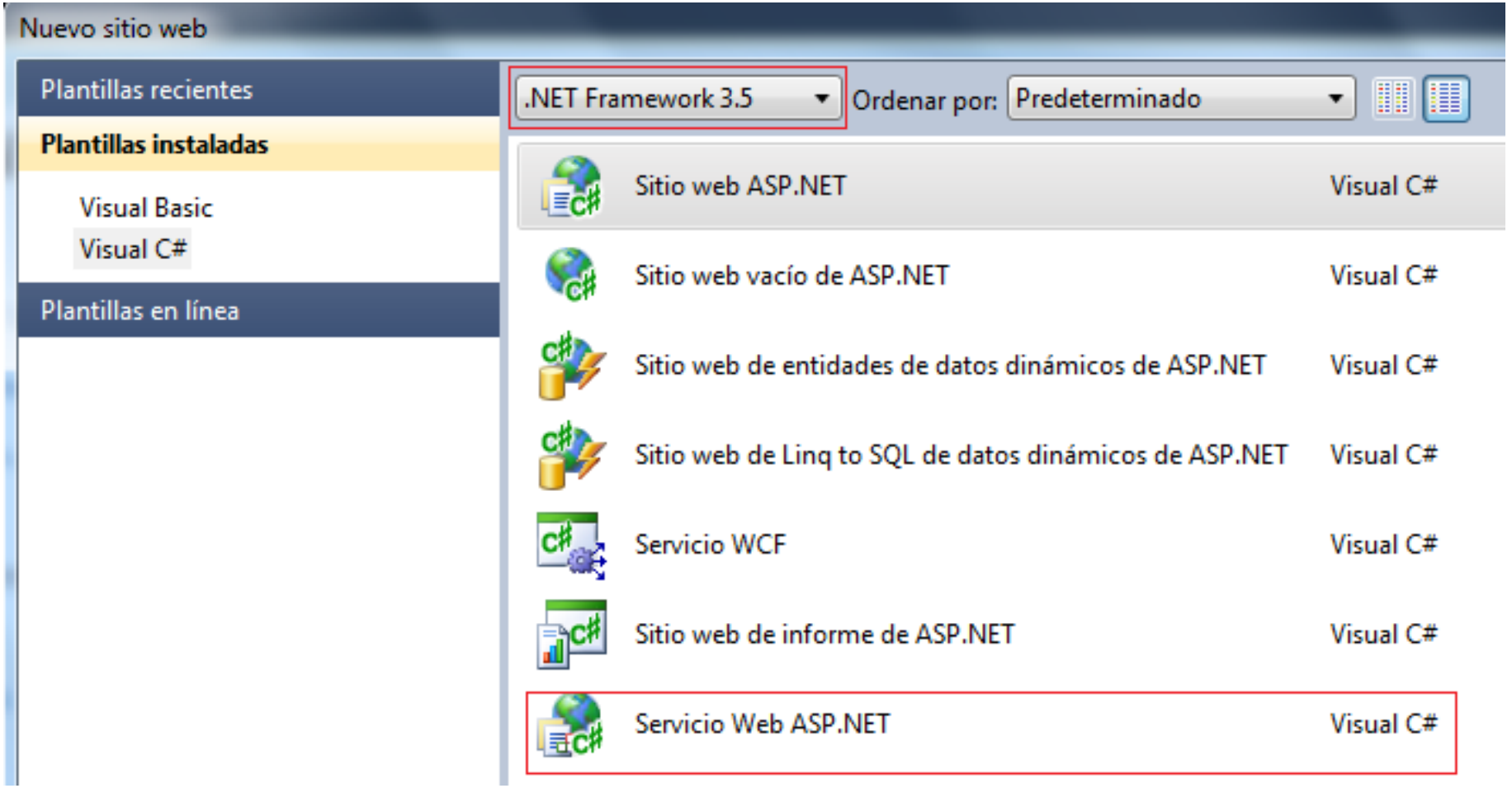
Implementación ASMX

Opciones HTTP

Opciones	ASMX
Datos	SOAP, XML, JSON
Verbos	POST, GET
Encabezados (Headers)	Incluye
Descripción	WebMethod
Cache	WebMethod
JS Proxy	ScriptMethod (habilitarlo)
Template URI	No incluye

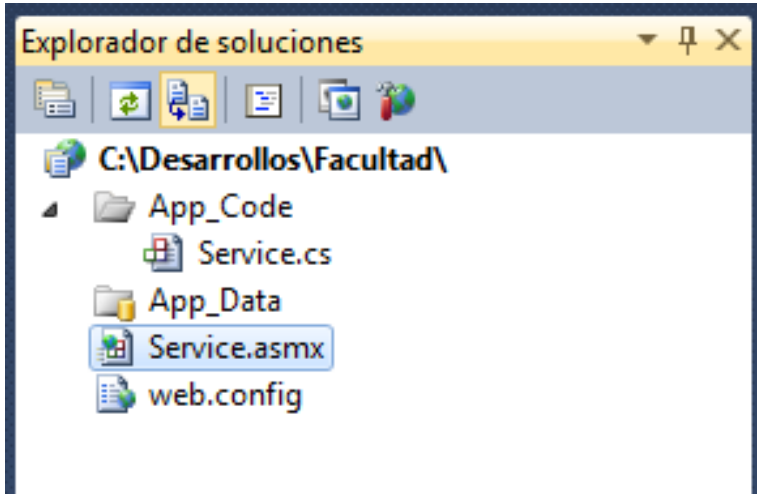
Implementación en .Net

Ejemplo .asmx - Creación



Implementación en .Net

Ejemplo .asmx - Creación



Implementación en .Net

Ejemplo .asmx - Creación

Dentro de Service.asmx

```
<%@ WebService Language="C#"
      CodeBehind="~/App_Code/Service.cs" Class="Service" %>
```

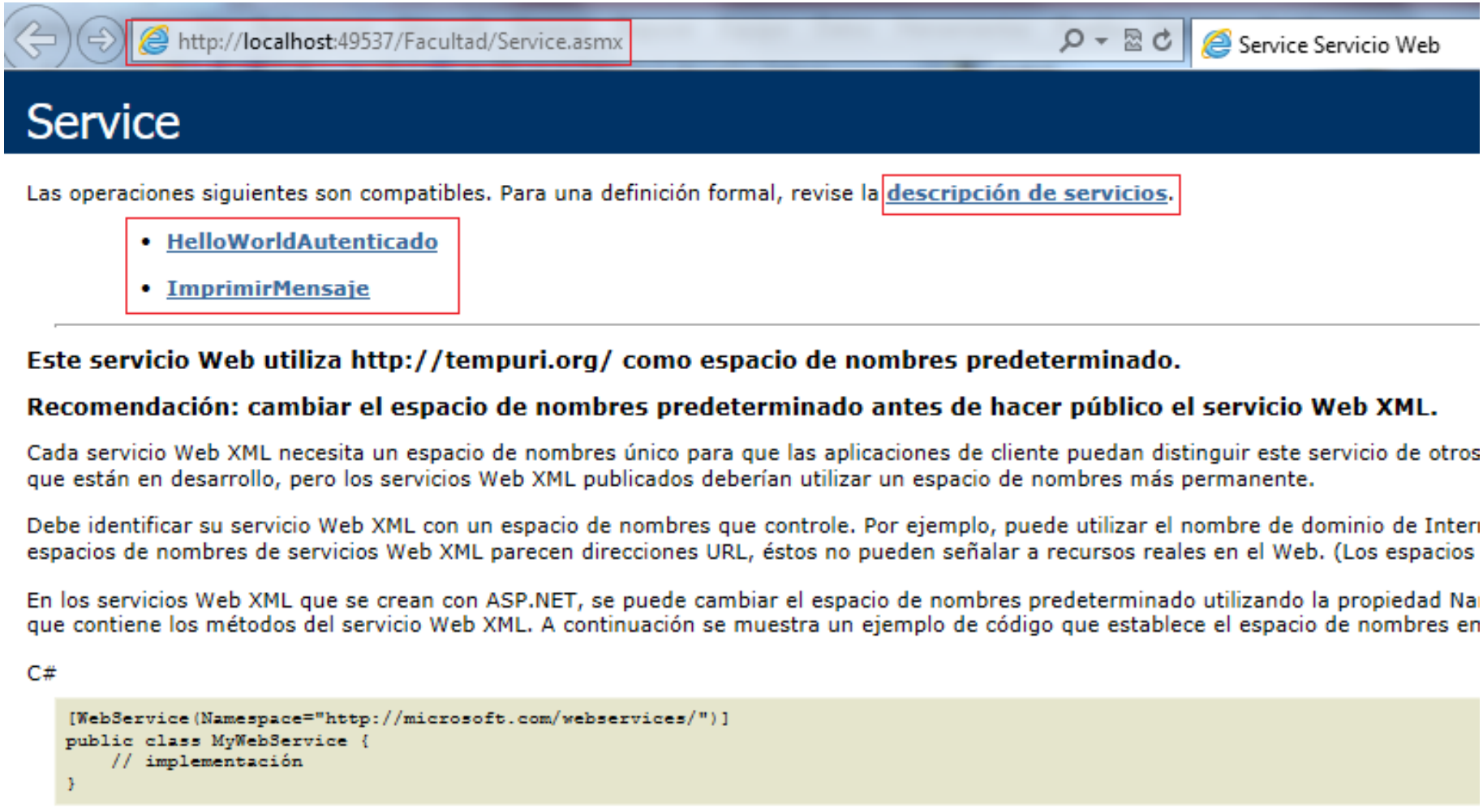
Implementación en .Net

Ejemplo .asmx.cs - Creación

```
using System;
using System.Web;
using System.Web.Services;

[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[System.Web.Script.Services.ScriptService]
public class Service : System.Web.Services.WebService
{
    public Service () {
        //Eliminar la marca de comentario de la línea siguiente si utiliza los componentes diseñados
        //InitializeComponent();
    }

    [WebMethod]
    public string ImprimirMensaje(string mensaje) {
        return mensaje;
    }

    [WebMethod(EnableSession=true)]
    public string HelloWorldAutenticado()
    {
        string usuario = HttpContext.Current.Session["Usuario"].ToString();

        if (usuario != String.Empty)
        {
            return usuario;
        }
        else
        {
            return "Usuario No Autenticado";
        }
    }
}
```

Implementación en .Net

Ejemplo .asmx - Ejecución



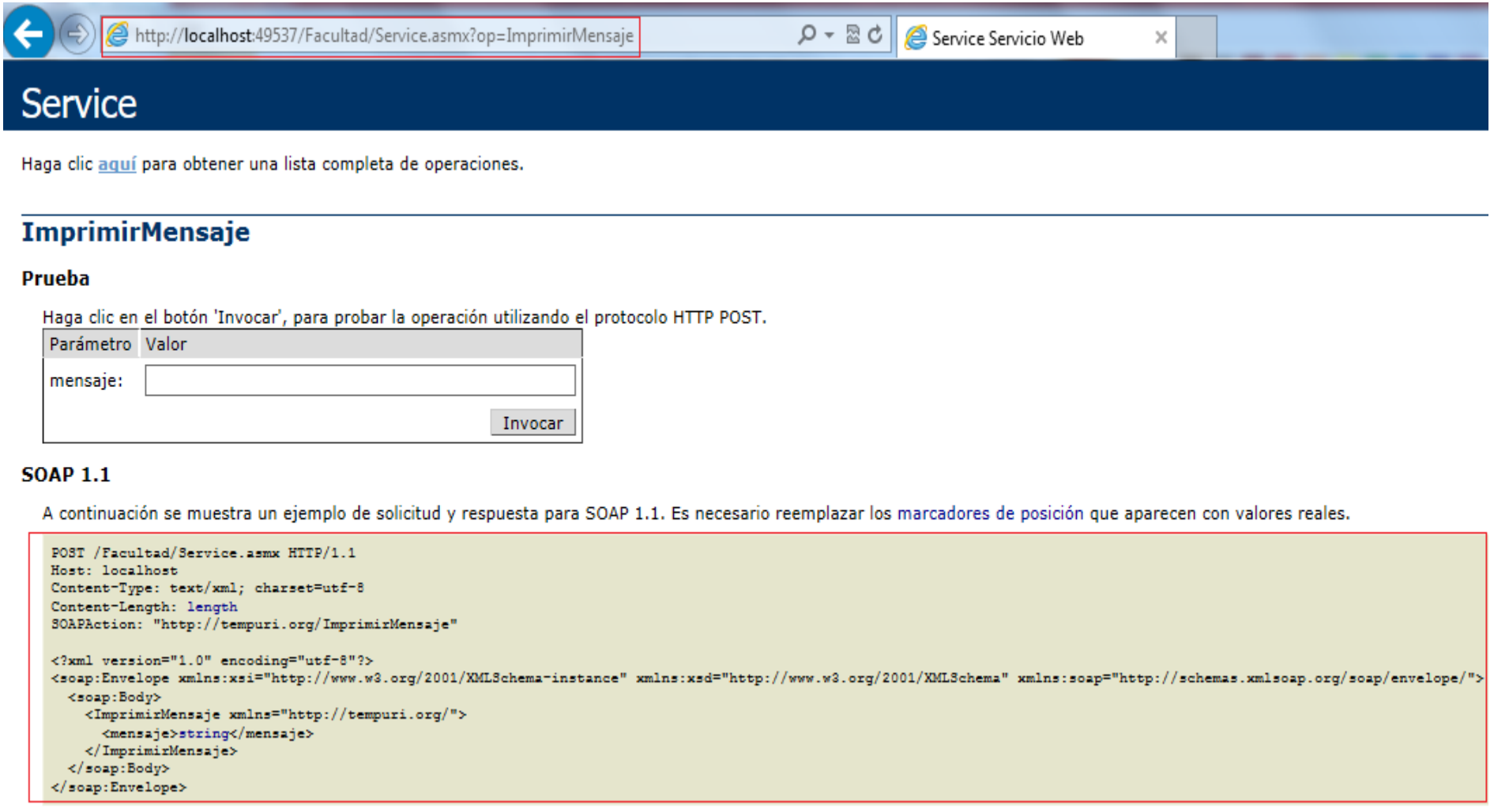
Implementación en .Net

Ejemplo .asmx - Ejecución


```
<?xml version="1.0" encoding="UTF-8"?>
- <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://tempuri.org/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://tempuri.org/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  - <wsdl:types>
    - <s:schema targetNamespace="http://tempuri.org/" elementFormDefault="qualified">
      - <s:element name="ImprimirMensaje">
        - <s:complexType>
          - <s:sequence>
              <s:element name="mensaje" type="s:string" maxOccurs="1" minOccurs="0"/>
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:element name="ImprimirMensajeResponse">
        - <s:complexType>
          - <s:sequence>
              <s:element name="ImprimirMensajeResult" type="s:string" maxOccurs="1" minOccurs="0"/>
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:element name="HelloWorldAutenticado">
          <s:complexType/>
        </s:element>
      - <s:element name="HelloWorldAutenticadoResponse">
        - <s:complexType>
          - <s:sequence>
              <s:element name="HelloWorldAutenticadoResult" type="s:string" maxOccurs="1" minOccurs="0"/>
            </s:sequence>
          </s:complexType>
        </s:element>
```

Implementación en .Net

Ejemplo .asmx - Ejecución



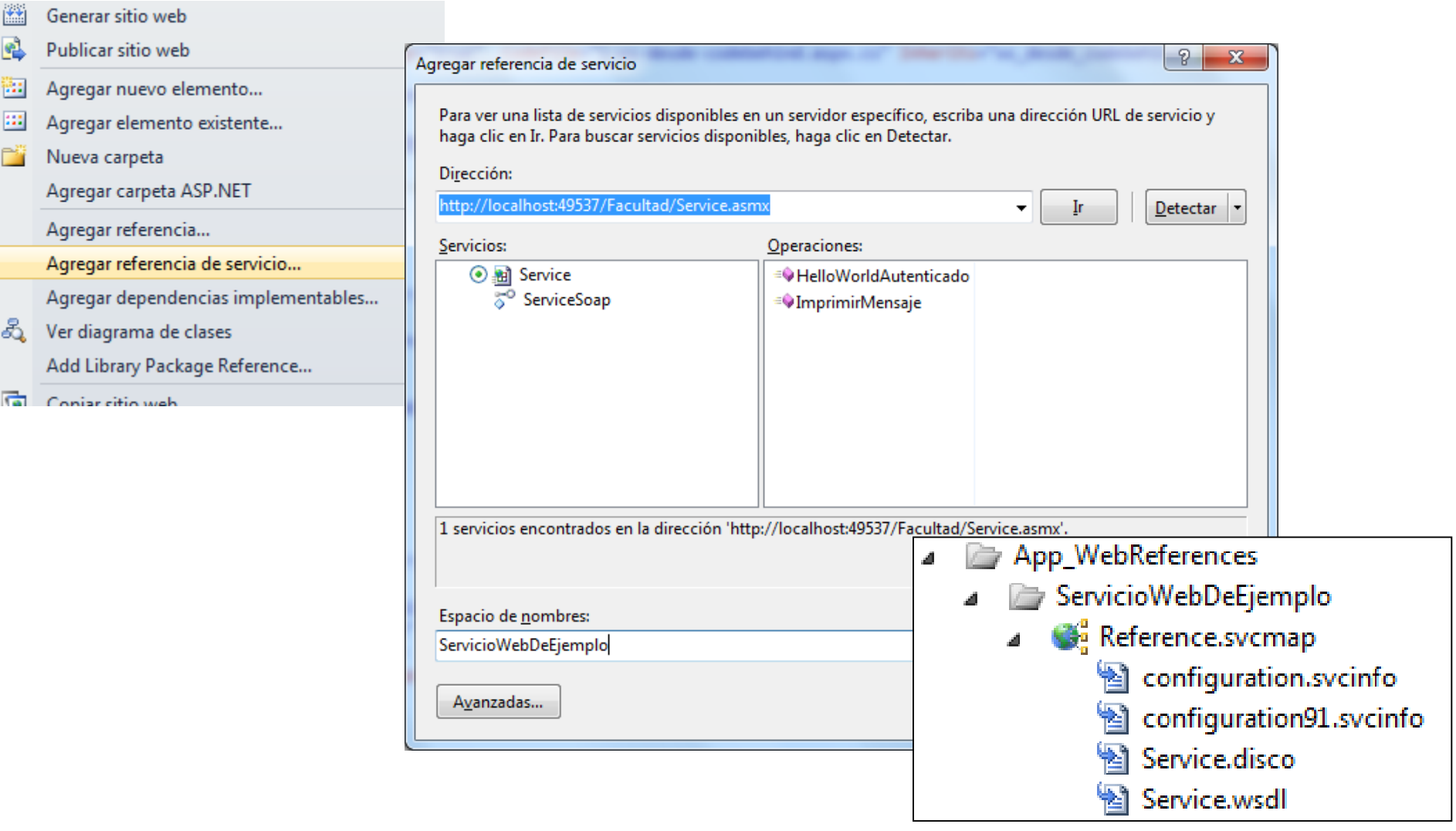
Implementación en .Net

Ejemplo .aspx - Ejecución


```
<?xml version="1.0" encoding="UTF-8"?>
<string xmlns="http://tempuri.org/">Hola Mundo</string>
```

Implementación en .Net

Ejemplo .aspx - Referencia a Servicio (Propio/Externo)



Implementación en .Net

Ejemplo .aspx.cs - Referencia a Servicio (Propio/Externo)

```
ServicioWebDeEjemplo.ServiceSoapClient servicio =
                    new ServicioWebDeEjemplo.ServiceSoapClient();

string mensaje = servicio.ImprimirMensaje("Hola Mundo");
```

Implementación en .Net

Web.Config - Referencia a Servicio (Propio/Externo)

```
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="ServiceSoap" closeTimeout="00:01:00" openTimeout="00:01:00"
        receiveTimeout="00:10:00" sendTimeout="00:01:00" allowCookies="false"
        bypassProxyOnLocal="false" hostNameComparisonMode="StrongWildcard"
        maxBufferSize="65536" maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
        messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
        useDefaultWebProxy="true">
        <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
          maxBytesPerRead="4096" maxNameTableCharCount="16384" />
        <security mode="None">
          <transport clientCredentialType="None" proxyCredentialType="None"
            realm="" />
          <message clientCredentialType="UserName" algorithmSuite="Default" />
        </security>
      </binding>
    </basicHttpBinding>
  </bindings>
  <client>
    <endpoint address="http://localhost:49537/Facultad/Service.asmx"
      binding="basicHttpBinding" bindingConfiguration="ServiceSoap"
      contract="ServicioWebDeEjemplo.ServiceSoap" name="ServiceSoap" />
  </client>
</system.serviceModel>
```

Implementación en Javascript / jQuery

Llamado desde Script de Javascript utilizando jQuery

```
$.support.cors = true;

$.ajax({
    type: "POST",
    url: "http://localhost:49537/Facultad/Service.asmx/ImprimirMensaje",
    data: "{'mensaje':'Hola Mundo!'}",
    async: true, //por defecto es true
    contentType: "application/json; charset=utf-8",
    dataType: "json",
    success: function (response) {
        alert(response.d);
    },
    error: function (mensajeError) {
        //cualquier error del lado servidor sale por este evento
        alert(mensajeError);
    }
});
```

Windows Communication Foundation (WCF)

¿Qué és WCF?

Es un marco de trabajo para la creación de aplicaciones orientadas a servicios

Aparece en .NET 3.0

Una aplicación WCF está compuesta por:

Clientes: Son aplicaciones que inician la comunicación.

Servicios: Son aplicaciones que esperan los mensajes de los clientes y responden a los mismos

Los mensajes son enviados entre **endpoints**. Un **endpoint** es un lugar donde un mensaje es enviado, o recibido, o ambos.

Un servicio expone uno o más **application endpoints**, y un cliente genera un **endpoint** compatible con uno de los **endpoints** de un servicio dado.

La combinación de un servicio y un cliente compatibles conforman un **communication stack**.

Windows Communication Foundation (WCF)

Características

- Orientado a Servicios
- Interoperable
- Admite Metadatos de Servicios
- Incluye Contratos de datos
- Seguridad
- Varios Transporte y codificaciones
- Mensajes confiables y duraderos
- Utilización de Colas
- Transaccionabilidad
- Compatible con AJAX y REST
- Extensibilidad

Windows Communication Foundation (WCF)

Opciones HTTP

Opciones	ASMX	WCF
Protocolo	HTTP	HTTP, TCP, MSMQ, etc.
Datos	SOAP, XML, JSON	SOAP, XML, JSON
Verbos	POST, GET	GET, POST, PUT, DELETE
Encabezados (Headers)	Incluye	Incluye
Descripción	WebMethod	No incluye
Cache	WebMethod	En Headers
JS Proxy	ScriptMethod (habilitarlo)	Por defecto
Template URI	No incluye	Incluye
Hosting	IIS	IIS, Proceso .NET, Servicio Windows

Windows Communication Foundation (WCF)

Pasos para Desarrollar un Servicio WCF

1) Definir el Contrato ([ServiceContract]): Se escribe la interfaz en un lenguaje de programación de .NET, agregando los distintos métodos ([OperationContract]) que serán incluidos en el contrato.

2) Implementar el Contrato (ServiceContract): Se escribe una clase mediante la cual se implemente la interfaz. Es posible establecer comportamientos (Concurrencia, exponer metadatos, etc.) a la definición del servicio usando el atributo **ServiceBehavior**.

3) Configurar el Servicio: Especificar los **endpoints** (lugar/mecanismo) y **metadata** del servicio, estos son definidos en un archivo de configuración de .NET (**Web.config** o **App.config**).

4) Diseñar una aplicación Hosting del servicio:

- **Web Host** dentro del IIS
- **Self-Host** dentro de cualquier proceso .NET
- **Managed Windows Services**
- **Windows Process Activation Service**.

5) Diseñar una aplicación cliente del servicio

Windows Communication Foundation (WCF)

IServicio.cs

```
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;

namespace Servicio_WCF_SOAP_REST
{
    [ServiceContract]
    public interface IServicio
    {
        [OperationContract]
        [WebGet(UriTemplate="/Empleados",ResponseFormat=WebMessageFormat.Xml )]
        Empleado[] GetEmpleados();
    }

    [DataContract]
    public class Empleado
    {
        [DataMember]
        public int ID { get; set; }
        [DataMember]
        public string NomyApel{ get; set; }
        [DataMember]
        public string DNI { get; set; }
    }
}
```

Windows Communication Foundation (WCF)

Web.config

```
….
  <system.serviceModel>
    <services>
      <service name=" Servicio_WCF_SOAP_REST"  behaviorConfiguration="ServBehave">
        <!--Endpoint for SOAP-->
        <endpoint
           address="soapService"
            binding="basicHttpBinding"
            contract=" Servicio_WCF_SOAP_REST.IServicio"/>
        <!--Endpoint for REST-->
        <endpoint
          address="XMLService"
           binding="webHttpBinding"
           behaviorConfiguration="restPoxBehavior"
           contract=" Servicio_WCF_SOAP_REST.IServicio"/>
      </service>
    </services>
…
```

Windows Communication Foundation (WCF)

Web.config

```
….
<behaviors>
    <serviceBehaviors>

        <behavior name="ServBehave">
            <serviceMetadata httpGetEnabled="true"/>
            <serviceDebug includeExceptionDetailInFaults="false"/>
        </behavior>
    </serviceBehaviors>

    <endpointBehaviors>
        <!--Behavior for the REST endpoint for Help enability-->
        <behavior name="restPoxBehavior">
          <webHttp helpEnabled="true"/>
        </behavior>
    </endpointBehaviors>
  </behaviors>

  <serviceHostingEnvironment aspNetCompatibilityEnabled="true"
multipleSiteBindingsEnabled="true"/>
 </system.serviceModel>
…
```

Desarrollador de Aplicaciones Web
Programación Web III

Departamento de Ingeniería e Investigaciones Tecnológicas

Muchas gracias

Ing. Mariano Juiz
Ing. Matias Paz Wasiuchnik
Ing. Pablo Nicolás Sanchez