EBook Gratis

APRENDIZAJE web-services

Free unaffiliated eBook created from
Stack Overflow contributors.

#web-services

Tabla de contenido

Acerca de 1

Capítulo 1: Comenzando con los servicios web 2

Observaciones 2

Examples 2

Instalación o configuración 2

Lado del servidor (servicios web de host) 2

Lado del cliente 3

¿Por qué utilizar servicios web? 3

Implementaciones de Java 3

Componentes relacionados con el servicio web 3

Capítulo 2: Llamar a los servicios web mediante programación utilizando C # .net y el desa 5

Introducción 5

Observaciones 5

Examples 5

Llamando al método GET simple 5

Servicio web de llamada con datos POST / POST Método 5

Llamar al servicio web con el método POST / POST de datos (Datos publicados en formato JSO 6

Llamada de servicio web con salida como objeto IEnumerator 6

Salida del servicio web en formato de lista o formato de tabla de datos 6

Hacer con fuerza el método GET OR POST 7

Capítulo 3: servicios web asp.net 8

Introducción 8

Sintaxis 8

Parámetros 8

Observaciones 8

Examples 11

Nota 11

Creditos 14

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: web-services

It is an unofficial and free web-services ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official web-services.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Comenzando con los servicios web

Observaciones

Un servicio web es un componente de aplicación web que permite la comunicación entre aplicaciones con fines de integración.

Los servicios web siguen una arquitectura cliente-servidor. Una aplicación "ofrece" un servicio web (servidor) y otras aplicaciones "consumen" el servicio web (cliente).

Se implementan sobre HTTP utilizando peticiones y respuestas.

Los principales tipos de servicios web son:

- JABÓN (**S** imple **O** bject **A** ccess **P** rotocol)
- REST **(RE** de presentación **S** tate **T** ransfer)

Examples

Instalación o configuración

Lado del servidor (servicios web de host)

Los servicios web deben instalarse y ejecutarse (implementarse) en un servidor web como componentes de aplicaciones web. Pueden ser parte de una aplicación más grande, o pueden implementarse solos, ya que pueden componer una aplicación completa.

Es responsabilidad del servidor reenviar una solicitud HTTP entrante a la aplicación implementada correspondiente, y la responsabilidad de la aplicación manejar la solicitud de acuerdo con:

- el verbo HTTP (GET, POST, PUT, DELETE, OPTIONS, HEAD, TRACE, CONNECT)
- la URL de solicitud

La aplicación utiliza la combinación de estos elementos para ubicar el componente de servicio web correspondiente que debe procesar la solicitud.

Una vez ubicado el servicio web, los parámetros de solicitud se utilizan como datos de entrada al servicio web. El servicio web es responsable de convertir los datos a los tipos de datos correctos y de establecer una convención con los clientes sobre la transmisión de diferentes tipos de datos.

El servicio web está procesando los datos de entrada y produce un conjunto de datos de salida. El conjunto de datos de salida está envuelto en una respuesta HTTP y se envía de vuelta al

remitente de la solicitud.

Lado del cliente

Un cliente debe preparar una solicitud HTTP, que cumpla con las reglas del servidor y enviarla al servidor. La respuesta que se recibirá contendrá los datos requeridos.

¿Por qué utilizar servicios web?

El uso de programas cliente de servicios web y un servidor pueden intercambiar información y colaborar para producir nuevos servicios y resultados, independientemente de su ubicación física y la tecnología en la que se basan. Solo deben cumplir con las especificaciones de nivel de aplicación.

La diferencia entre el uso de servicios web y el servicio web-HTML (navegación) es principalmente que los servicios web están enfocados y especializados en procesar y convertir tipos de datos para producir resultados estructurados, que se pueden usar para llamadas de procedimientos remotos. El servicio de Web-HTML es más sobre el servicio de recursos rendibles / descargables.

El intercambio de resultados de procesos mediante servicios web facilita:

- integración de aplicaciones
- separación de intereses
- Arquitecturas de aplicaciones distribuidas / descentralizadas.

Implementaciones de Java

En Java los servicios web se implementan como servlets. Los marcos de servicios web más populares están implementando un servlet que necesita ser mapeado con una URL. Ejemplos de marcos:

- Eje
- CXF
- Jersey

Componentes relacionados con el servicio web

1. WSDL (lenguaje de descripción de servicios web)
2. UDDI (Descripción Universal de Descubrimiento e Integración)
3. SOAP (Protocolo simple de acceso a objetos)

Lea Comenzando con los servicios web en línea: https://riptutorial.com/es/web-services/topic/5309/comenzando-con-los-servicios-web

Capítulo 2: Llamar a los servicios web mediante programación utilizando C # .net y el desarrollo de aplicaciones Xamarin

Introducción

Aquí veremos Pro-grammatically llamando y utilizando servicios web en ASP.Net C #. Para el propósito, deberá descargar el siguiente ddl, que le proporciona muchas funciones. Descargue ImportJson desde https://drive.google.com/open?id=0B-2bGoHKJvnOckdPUHVjdFZTcFU

Este artículo es muy útil para aquellos de ustedes que van a desarrollar un proyecto utilizando los servicios web de ASP.NET C # / servicios web API. Este artículo también es útil para aquellos que están desarrollando un proyecto usando Xamarin: Desarrollo de aplicaciones móviles

Observaciones

Debes dar referencia de ImportJson dll y restsharp ddl. ImportJson se puede descargar desde aquí https://drive.google.com/open?id=0B-2bGoHKJvnOckdPUHVjdFZTcFU y restsharp.dll obtendrá de internet

Cualquier sugerencia / contacto, por favor tenga en cuenta **akhandagale65@gmail.com**

Examples

Llamando al método GET simple

```
/// <summary>
    /// Simple Get method
    /// </summary>
    /// <returns> Json formated data </returns>
    public string GetJsonData1()
    {
        IOperations _Obj = ClsOperations.GetOperations();
        string url = "http://1.2.3.4:1234/Services/rest/CallService/WebRequest/";
        string jsonResult = _Obj.GetJsonResult(url);
        return jsonResult;
    }
```

Servicio web de llamada con datos POST / POST Método

```
        /// <summary>
    /// Post Method with input parameter
    /// </summary>
    /// <returns> Json formated data </returns>
    public string GetJsonData2()
```

```
    {
        IOperations _Obj = ClsOperations.GetOperations();
        string url = "http://1.2.3.4:1234/Services/rest/CallService/WebRequest/";
        Dictionary<string, object> objDec = new Dictionary<string, object>();
        objDec.Add("@FirstParameter", "Value1");
        objDec.Add("@SecondParameter", "Value2");
        objDec.Add("@ThirdParameter", "Value3");
        string jsonResult = _Obj.GetJsonResult(url, objDec);
        return jsonResult;
    }
```

Llamar al servicio web con el método POST / POST de datos (Datos publicados en formato JSON)

```
/// <summary>
    /// Post Method with Input/ data to post in JSON format
    /// </summary>
    /// <returns> Json formated data </returns>
    public string GetJsonData3()
    {
        IOperations _Obj = ClsOperations.GetOperations();
        string url = "http://1.2.3.4:1234/Services/rest/CallService/WebRequest/";
        string inputjson = "{\"@FirstParameter\": \"Value1\",\"@SecondParameter\":
\"Value2\",\"@ThirdParameter\": \"Value3\"}";
        string jsonResult = _Obj.GetJsonResult(url, null,inputjson );
        return jsonResult;
    }
```

Llamada de servicio web con salida como objeto IEnumerator

```
/// <summary>
    /// Post Method with Input/ data to post in JSON format Or you can send dictionary as
shown in previous methods
    /// </summary>
    /// <returns> Json formated data </returns>
    public void  GetJsonData4()
    {
        IOperations _Obj = ClsOperations.GetOperations();
        string url = "http://1.2.3.4:1234/Services/rest/CallService/WebRequest/";
        string inputjson = "{\"@FirstParameter\": \"Value1\",\"@SecondParameter\":
\"Value2\",\"@ThirdParameter\": \"Value3\"}";
        string jsonResult = _Obj.GetJsonResult(url, null, inputjson);
        IEnumerator objIEnumerator = _Obj.GetJsonEnumerableResult(jsonResult);
        // you can perform further operations on it

    }
```

Salida del servicio web en formato de lista o formato de tabla de datos

```
    /// <summary>
        /// Post Method with Input/ data to post in JSON format Or you can send dictionary as
shown in previous methods
        /// </summary>
        /// <returns> Json formated data </returns>
        public DataTable  GetJsonData6()
```

```
        {
            IOperations _Obj = ClsOperations.GetOperations();
            string url = "http://1.2.3.4:1234/Services/rest/CallService/WebRequest/";
            string inputjson = "{\"@FirstParameter\": \"Value1\",\"@SecondParameter\":
\"Value2\",\"@ThirdParameter\": \"Value3\"}";
            IEnumerator objIEnumerator = _Obj.GetJsonEnumerableResult(url, null, inputjson);
            // you can perform further operations on it

            // If you want to convert it in Datatable / List

            List<ClsMyPropertyClass> lst = new List<ClsMyPropertyClass>();
            while (objIEnumerator.MoveNext())
            {

lst.Add(Newtonsoft.Json.JsonConvert.DeserializeObject<ClsLineEDoDetails>(objIEnumerator.Current.ToStrin

            }
// Upto this you will get List , and you can perform operations on it

            // Now if youu want result in datatable, here i written function for List to
datatable conversion

            return CommonServiceCall.ToDataTable(lst);

        }
```

Hacer con fuerza el método GET OR POST

```
/* By Default if you send only url then automatically it will recognize as GET Method and if
service having parameters with, Then automatically will convert to POST Method. But I observed
some of the services having only URL but are POST Type. For the purpose you can forcefully
make the method as you want. As bellow:  */
        /// <summary>
        /// If you want make the service call GET OR POST forcefully then
        /// </summary>
        /// <returns> Json formated data </returns>
        public void GetJsonData5()
        {
            IOperations _Obj = ClsOperations.GetOperations();
            string url = "http://1.2.3.4:1234/Services/rest/CallService/WebRequest/";
            string inputjson = "{\"@FirstParameter\": \"Value1\",\"@SecondParameter\":
\"Value2\",\"@ThirdParameter\": \"Value3\"}";
string  _result =   _ Obj.GetJsonResult(url, null, inputjson, ServiceType.POST);;
                     }
```

Lea Llamar a los servicios web mediante programación utilizando C # .net y el desarrollo de aplicaciones Xamarin en línea: https://riptutorial.com/es/web-services/topic/9689/llamar-a-los-servicios-web-mediante-programacion-utilizando-c-sharp--net-y-el-desarrollo-de-aplicaciones-xamarin

Capítulo 3: servicios web asp.net

Introducción

El servicio web es una aplicación que está diseñada para interactuar directamente con otras aplicaciones a través de Internet. En sentido simple, los servicios web son medios para interactuar con objetos a través de Internet. Los consumidores de `Web serivce` pueden invocar llamadas de método en objetos remotos utilizando `SOAP` y `HTTP` través de la web. `WebService` es independiente del idioma y los servicios web se comunican mediante el uso de protocolos web estándar y formatos de datos, como HTTP, XML, SOAP.

Sintaxis

1. **SOAP / WSDL**

   [Sintaxis: http: // 1111: 22 / HelloWorld]

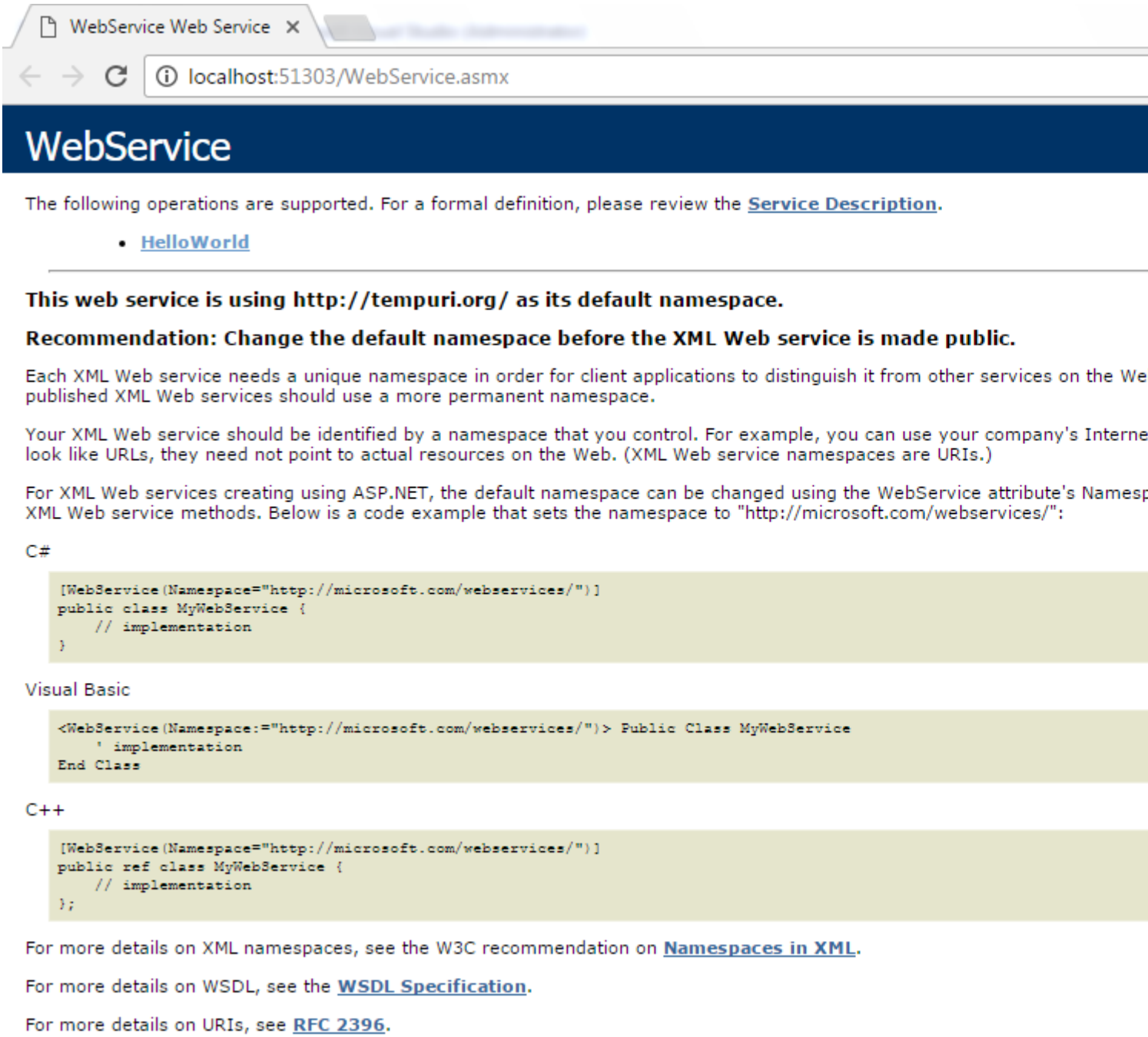
   [Respuesta: WDSL]

Parámetros

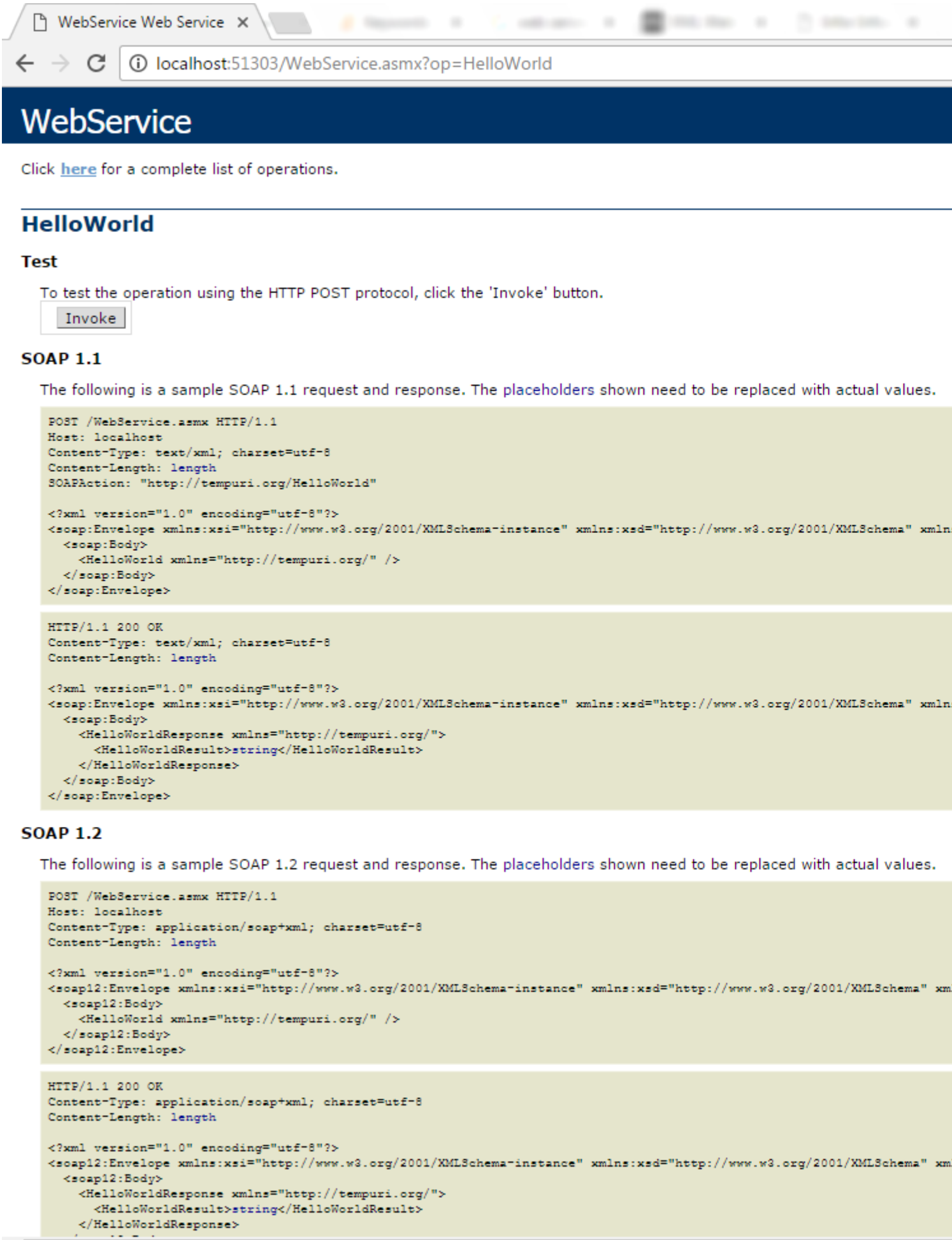
Parámetros	Descripciones
PathParam	Enlaza el parámetro pasado al método a un valor en la ruta.
QueryParam	Enlaza el parámetro pasado al método a un parámetro de consulta en la ruta.
MatrixParam	Enlaza el parámetro pasado al método a un parámetro de matriz HTTP en la ruta.
HeaderParam	Enlaza el parámetro pasado al método a un encabezado HTTP.
CookieParam	Enlaza el parámetro pasado al método a una cookie.
FormParam	Enlaza el parámetro pasado al método a un valor de formulario.
Valor por defecto	Asigna un valor predeterminado a un parámetro pasado al método.
Contexto	Contexto del recurso, por ejemplo, HTTPRequest como contexto.

Observaciones

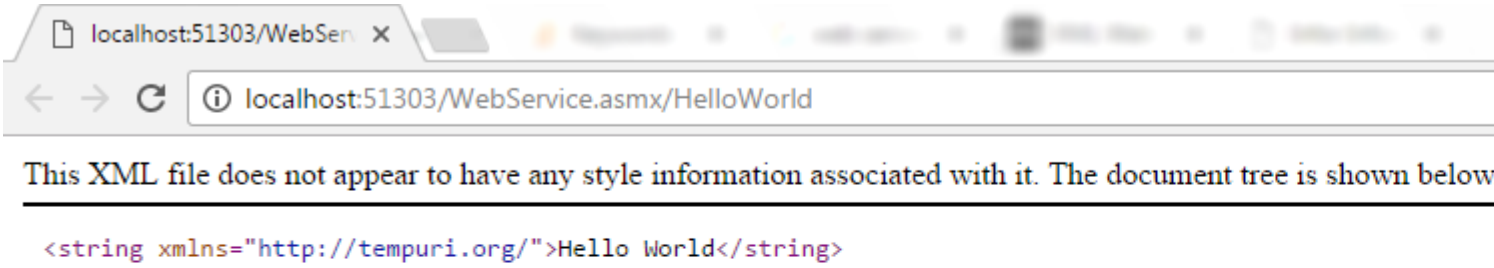
Ahora ejecuta la aplicación que se ve como sigue.



Ahora, en el ejemplo anterior, vemos nuestro método de creación en el archivo webservice.cs, así que haga clic en ese método y proporcione los valores de entrada y haga clic en el enlace "invocar" como en.


WebService Web Service
localhost:51303/WebService.asmx?op=HelloWorld
WebService
Click here for a complete list of operations.
HelloWorld
Test
To test the operation using the HTTP POST protocol, click the 'Invoke' button.
Invoke
SOAP 1.1
The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.
POST /WebService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/HelloWorld"
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmln
<soap:Body>
<HelloWorld xmlns="http://tempuri.org/" />
</soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmln
<soap:Body>
<HelloWorldResponse xmlns="http://tempuri.org/">
<HelloWorldResult>string</HelloWorldResult>
</HelloWorldResponse>
</soap:Body>
</soap:Envelope>
SOAP 1.2
The following is a sample SOAP 1.2 request and response. The placeholders shown need to be replaced with actual values.
POST /WebService.asmx HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xm
<soap12:Body>
<HelloWorld xmlns="http://tempuri.org/" />
</soap12:Body>
</soap12:Envelope>
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xm
<soap12:Body>
<HelloWorldResponse xmlns="http://tempuri.org/">
<HelloWorldResult>string</HelloWorldResult>
</HelloWorldResponse>

La salida será la siguiente



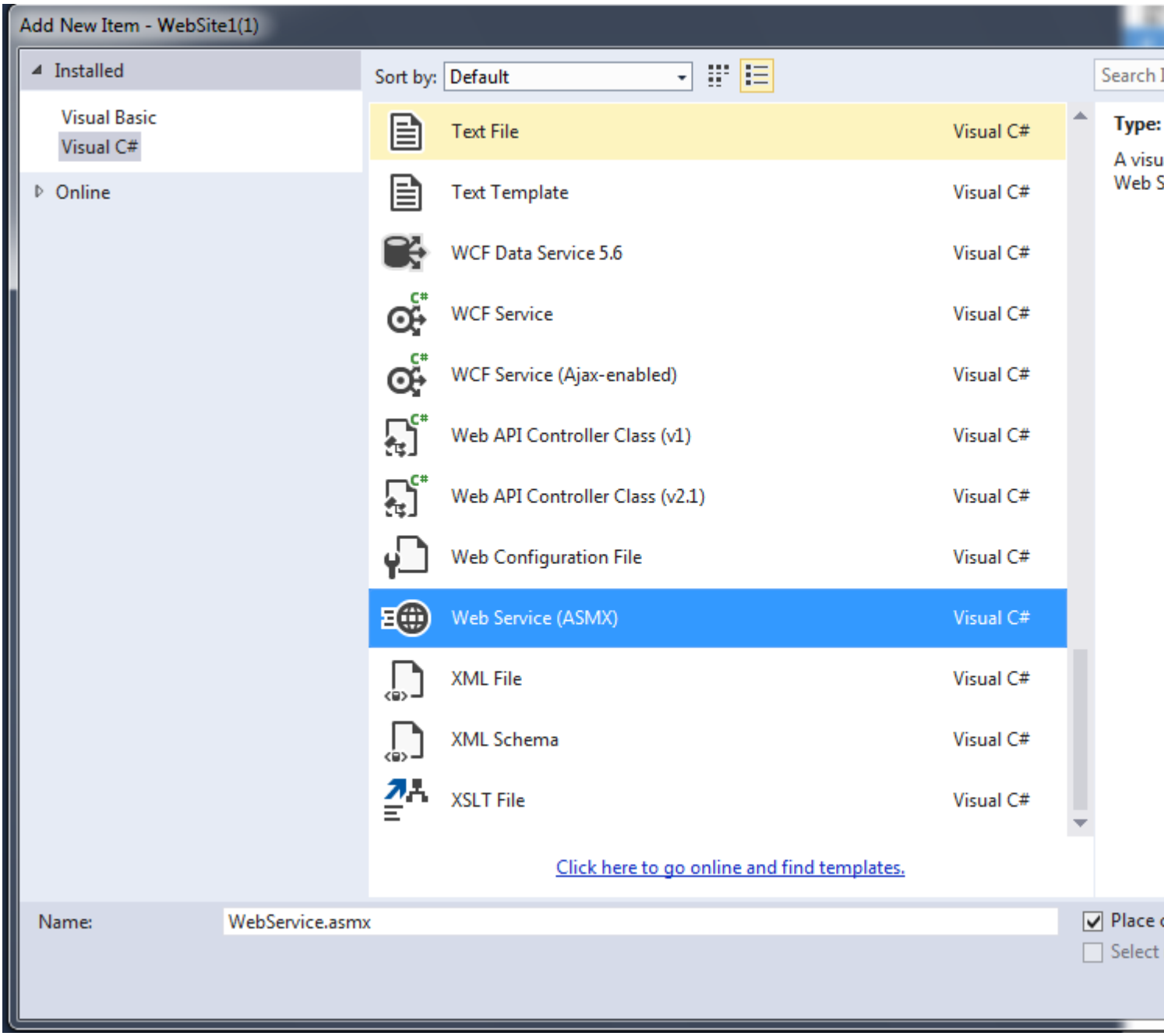
Examples

Nota

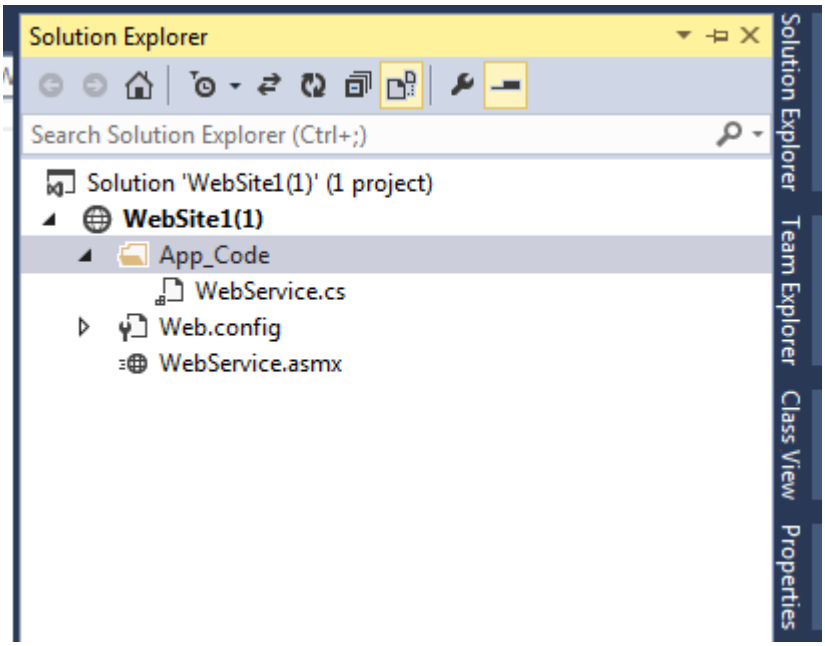
Si observa detenidamente eso, no hay una plantilla de servicio web por separado en .Framework 2010 como se ve en 2008 mientras agrega un proyecto o sitio web debido a WCF.

Así que comencemos a usar una forma diferente de agregar un servicio web usando una plantilla

1. "Inicio" - "Todos los programas" - "Microsoft Visual Studio 2010"
2. "Archivo" - "Nuevo proyecto" - "C #" - "Aplicación web vacía" (para evitar agregar una página maestra)
3. Proporcione al sitio web un nombre como "agetodays" u otro que desee y especifique la ubicación
4. Luego haga clic con el botón derecho en el Explorador de soluciones - "Agregar nuevo elemento": verá las plantillas de servicio web



Seleccione la plantilla de servicio web y haga clic en el botón Agregar. luego, después de eso, el Explorador de soluciones tiene el siguiente aspecto



Luego, abra la clase Webservice.cs y escriba el siguiente método seguido del atributo [webMethod] como en.

```
[WebMethod]
    public string HelloWorld() {
        return "Hello World";
    }
```

Lea servicios web asp.net en línea: https://riptutorial.com/es/web-services/topic/8341/servicios-web-asp-net

Creditos

S. No	Capítulos	Contributors
1	Comenzando con los servicios web	Community, Naga G, sanastasiadis
2	Llamar a los servicios web mediante programación utilizando C # .net y el desarrollo de aplicaciones Xamarin	Amol Khandagale
3	servicios web asp.net	Denish Parvadia