eBook Gratuit

APPRENEZ web-services

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#web-services

Table des matières

À propos

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

Chapitre 1: Démarrer avec les services Web

Remarques

Un service Web est un composant d'application Web qui permet la communication entre les applications à des fins d'intégration.

Les services Web suivent une architecture client-serveur. Une application "offre" un service web (serveur) et d'autres applications "consomment" le web-service (client).

Ils sont implémentés sur HTTP en utilisant des requêtes et des réponses.

Les principaux types de services Web sont les suivants:

- SOAP **(S** œuvre **O** bjet **A** ccess **P** ROTOCOLE)
- REST **(RE** présentation **S** tate **T** ransfert)

Examples

Installation ou configuration

Côté serveur (services Web hôte)

Les services Web doivent être installés et exécutés (déployés) sur un serveur Web en tant que composants d'application Web. Ils peuvent faire partie d'une application plus grande ou être déployés seuls, car ils peuvent composer une application complète.

Il est de la responsabilité du serveur de transmettre une requête HTTP entrante à l'application déployée correspondante, et la responsabilité de l'application de traiter la demande en fonction de:

- le verbe HTTP (GET, POST, PUT, DELETE, OPTIONS, HEAD, TRACE, CONNECT)
- l'URL de la demande

L'application utilise la combinaison de ces éléments pour localiser le composant de service Web correspondant qui doit traiter la demande.

Une fois le service Web localisé, les paramètres de la demande sont utilisés comme données d'entrée dans le service Web. Le service Web est chargé de convertir les données en types de données corrects et d'établir une convention avec les clients concernant la transmission de différents types de données.

Le service Web traite les données d'entrée et génère un jeu de données de sortie. Le jeu de données en sortie est encapsulé dans une réponse HTTP et il est renvoyé à l'expéditeur de la demande.

Côté client

Un client doit préparer une requête HTTP conforme aux règles du serveur et l'envoyer au serveur. La réponse qui sera reçue contiendra les données requises.

Pourquoi utiliser les services Web?

L'utilisation de programmes clients de services Web et d'un serveur permet d'échanger des informations et de collaborer pour produire de nouveaux services et résultats, quel que soit leur emplacement physique et la technologie sur laquelle ils sont construits. Ils doivent uniquement respecter les spécifications du niveau d'application.

La différence entre l'utilisation de services Web et la diffusion Web-HTML (navigation) réside principalement dans le fait que les services Web sont ciblés et spécialisés dans le traitement et la conversion de types de données pour produire des résultats structurés, utilisables pour les appels de procédures distants. Web-HTML-desservant est plus sur le service des ressources rendables / téléchargeables.

L'échange des résultats de processus à l'aide de services Web facilite:

- intégration d'applications
- séparation des préoccupations
- architectures d'applications distribuées / décentralisées

Implémentations Java

En Java, les services Web sont implémentés en tant que servlets. Les infrastructures de services Web les plus populaires implémentent un servlet qui doit être mappé avec une URL. Exemples de frameworks:

- Axe
- CXF
- Jersey

Composants liés au service Web

1. WSDL (Langage de description du service Web)
2. UDDI (Universal Description Discovery et Intégration)
3. SOAP (Simple Object Access Protocol)

Lire Démarrer avec les services Web en ligne: https://riptutorial.com/fr/web-services/topic/5309/demarrer-avec-les-services-web

Chapitre 2: Appel des services Web par programmation à l'aide du développement d'applications C # .net et Xamarin

Introduction

Nous verrons ici l'appel et l'utilisation pro-grammatical des services Web dans ASP.Net C #. Pour le but que vous aurez besoin de télécharger après ddl qui vous fournit de nombreuses fonctions. Téléchargez ImportJson depuis https://drive.google.com/open?id=0B-2bGoHKJvnOckdPUHVjdFZTcFU

Cet article est très utile pour ceux d'entre vous qui vont développer un projet à l'aide des services Web / Web API Services ASP.NET C #. Cet article est également utile pour ceux qui développent un projet utilisant Xamarin: Développement d'applications mobiles

Remarques

Vous devez donner une référence à ImportJson dll et restsharp ddl. ImportJson peut être téléchargé à partir d'ici https://drive.google.com/open?id=0B-2bGoHKJvnOckdPUHVjdFZTcFU Et restsharp.dll sera disponible à partir d'Internet

Toute suggestion / contact, s'il vous plaît noter **akhandagale65@gmail.com**

Examples

Appel de la méthode 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;
    }
```

Appel du service Web avec la méthode POST / POST de données

```
        /// <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;
    }
```

Appel du service Web avec la méthode Data POST / POST (publication de données au format 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;
    }
```

Appel de service Web avec sortie en tant qu'objet 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

    }
```

Sortie du service Web au format liste ou au format DataTable

```
    /// <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);

        }
```

Forcer avec force la méthode 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);;
                     }
```

Lire Appel des services Web par programmation à l'aide du développement d'applications C # .net et Xamarin en ligne: https://riptutorial.com/fr/web-services/topic/9689/appel-des-services-web-par-programmation-a-l-aide-du-developpement-d-applications-c-sharp--net-et-xamarin

Chapitre 3: asp.net web-services

Introduction

Web Service est une application conçue pour interagir directement avec d'autres applications sur Internet. En termes simples, les services Web sont des moyens d'interagir avec des objets sur Internet. Les `Web serivce` peuvent appeler des appels de méthode sur des objets distants en utilisant `SOAP` et `HTTP` sur le Web. `WebService` est indépendant de la langue et les services Web communiquent en utilisant des protocoles Web standard et des formats de données, tels que - HTTP - XML - SOAP

Syntaxe

1. **SOAP / WSDL**

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

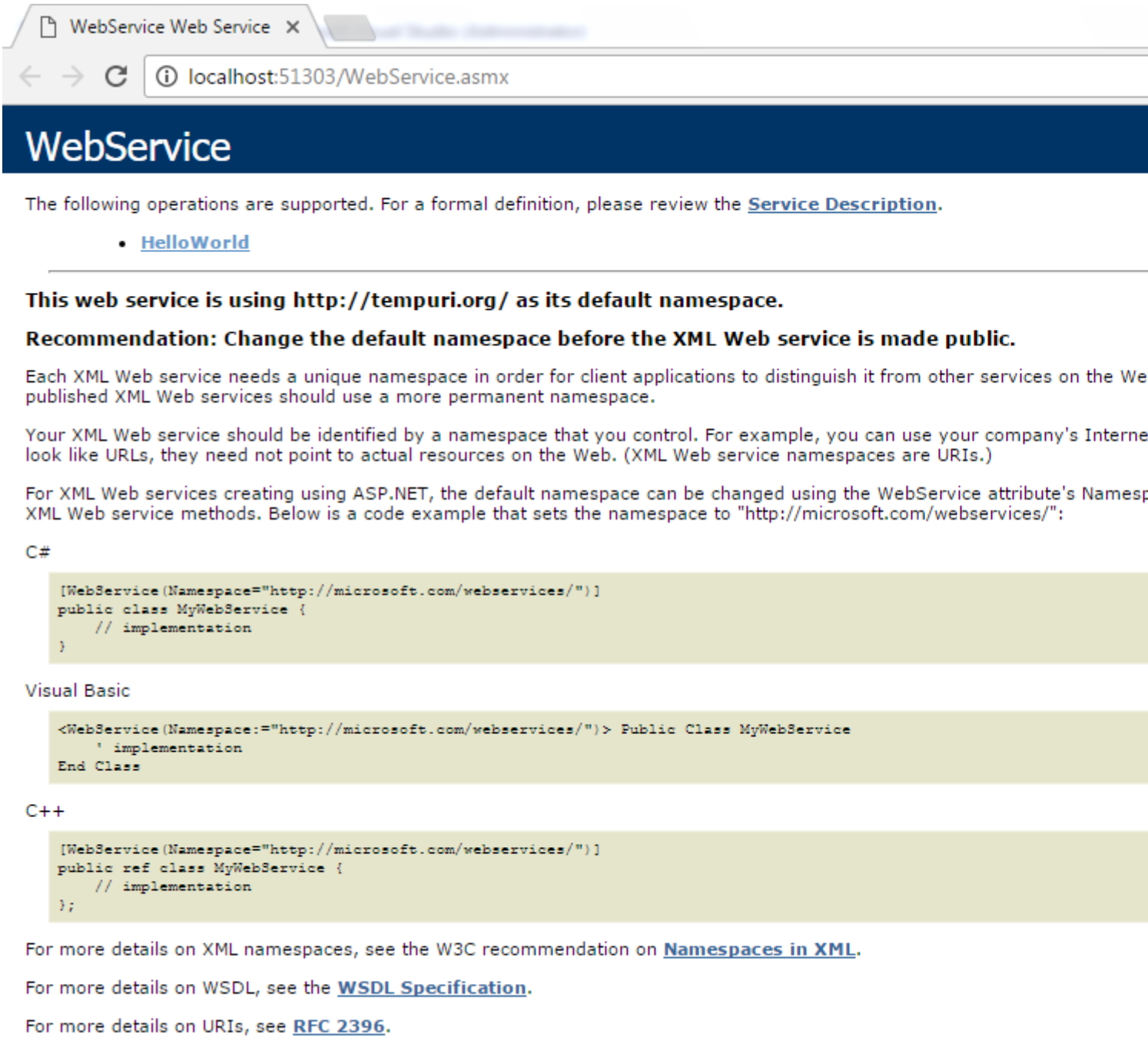
   [Réponse: WDSL]

Paramètres

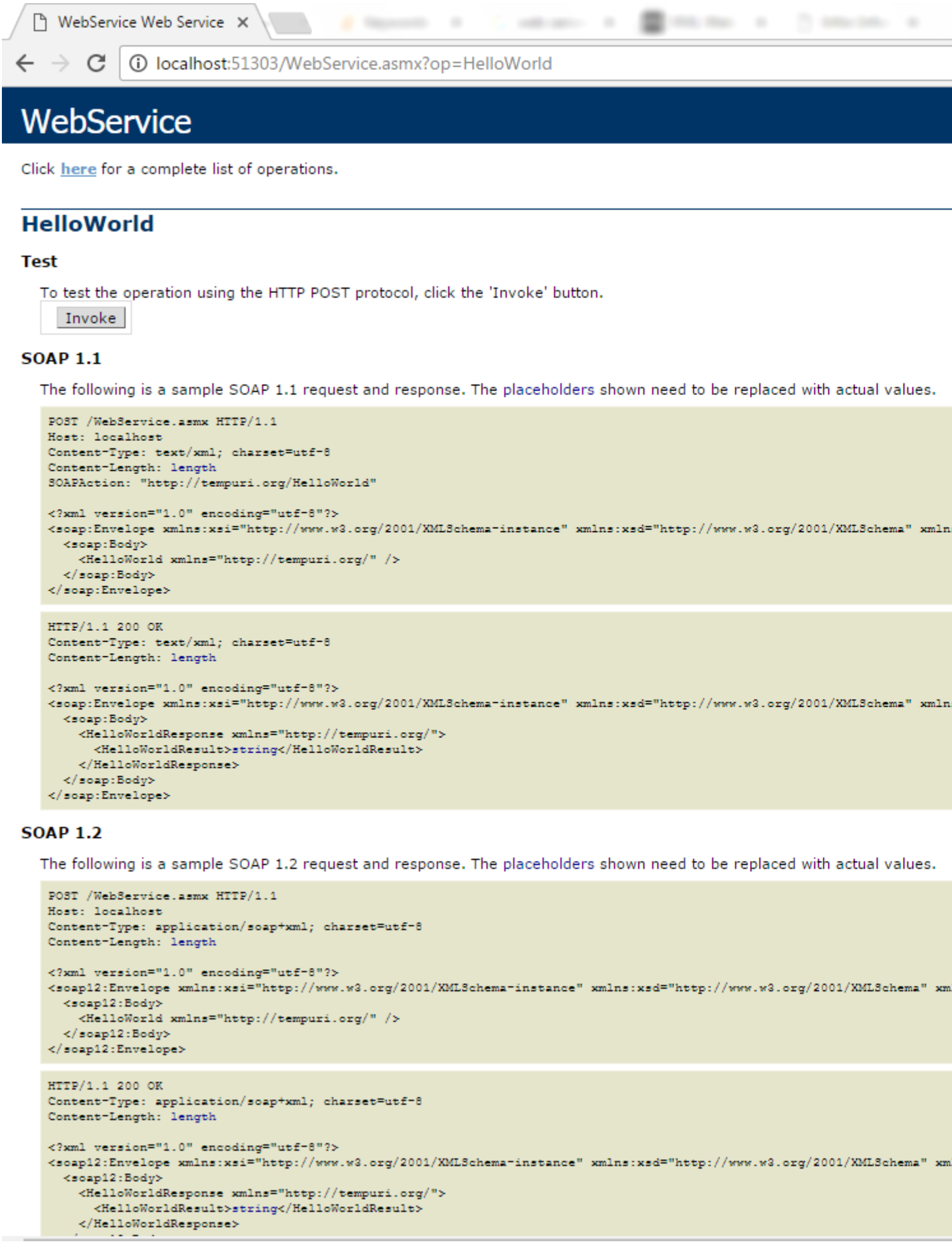
Paramètres	Les descriptions
PathParam	Lie le paramètre passé à la méthode à une valeur dans le chemin.
QueryParam	Lie le paramètre passé à la méthode à un paramètre de requête dans le chemin.
MatrixParam	Lie le paramètre passé à la méthode à un paramètre de matrice HTTP dans le chemin.
HeaderParam	Lie le paramètre passé à la méthode à un en-tête HTTP.
CookieParam	Lie le paramètre passé à la méthode à un cookie.
FormParam	Lie le paramètre passé à la méthode à une valeur de formulaire.
Valeur par défaut	Assigne une valeur par défaut à un paramètre passé à la méthode.
Le contexte	Contexte de la ressource par exemple HTTPRequest en tant que contexte.

Remarques

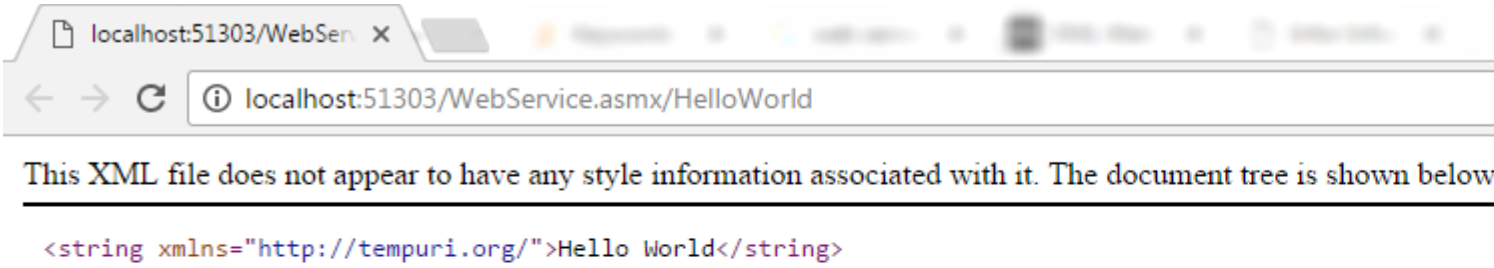
Exécutez maintenant l'application qui ressemble à ceci.



Maintenant, dans la partie ci-dessus, nous voyons notre méthode que nous avons créée dans le fichier webservice.cs, alors cliquez sur cette méthode et fournissez des valeurs d'entrée et cliquez sur le lien "invoke" comme dans.


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"
<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"
<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"
<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"
<soap12:Body>
<HelloWorldResponse xmlns="http://tempuri.org/">
<HelloWorldResult>string</HelloWorldResult>
</HelloWorldResponse>

La sortie sera comme suit



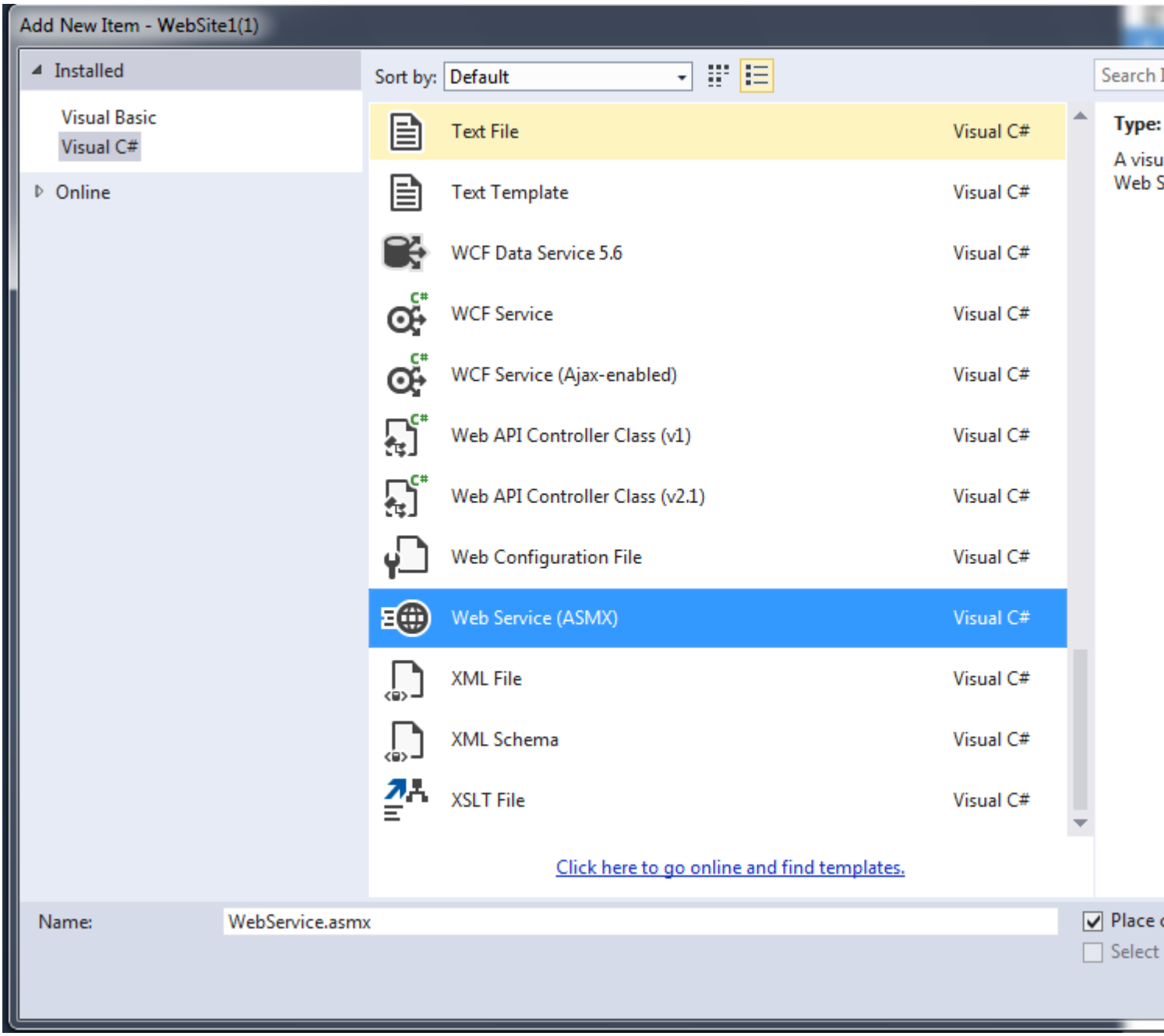
Examples

Remarque

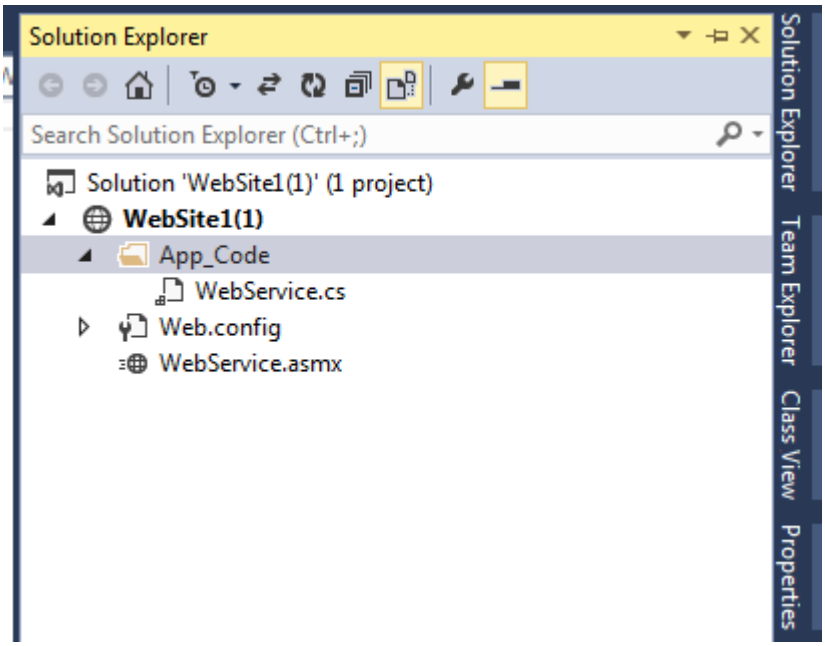
Si vous observez cela de près, il n'y a pas de modèle de service Web séparé dans .Framework 2010 comme vous le voyez en 2008 lors de l'ajout d'un projet ou d'un site Web, à cause de WCF.

Commençons donc à utiliser une autre façon d'ajouter un service Web à l'aide d'un modèle

1. "Démarrer" - "Tous les programmes" - "Microsoft Visual Studio 2010"
2. "Fichier" - "Nouveau projet" - "C #" - "Application Web vide" (pour éviter d'ajouter une page maître)
3. Fournir au site Web un nom tel que "agetodays" ou autre que vous souhaitez et spécifier l'emplacement
4. Cliquez ensuite avec le bouton droit sur Solution Explorer - "Ajouter un nouvel élément" - vous voyez les modèles de service Web



Sélectionnez Modèle de service Web et cliquez sur le bouton Ajouter. puis après que l'explorateur de solutions ressemble à ceci.



Ensuite, ouvrez la classe Webservice.cs et écrivez la méthode suivante suivie de l'attribut [webMethod] comme dans.

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

Lire asp.net web-services en ligne: https://riptutorial.com/fr/web-services/topic/8341/asp-net-web-services

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec les services Web	Community, Naga G, sanastasiadis
2	Appel des services Web par programmation à l'aide du développement d'applications C # .net et Xamarin	Amol Khandagale
3	asp.net web-services	Denish Parvadia