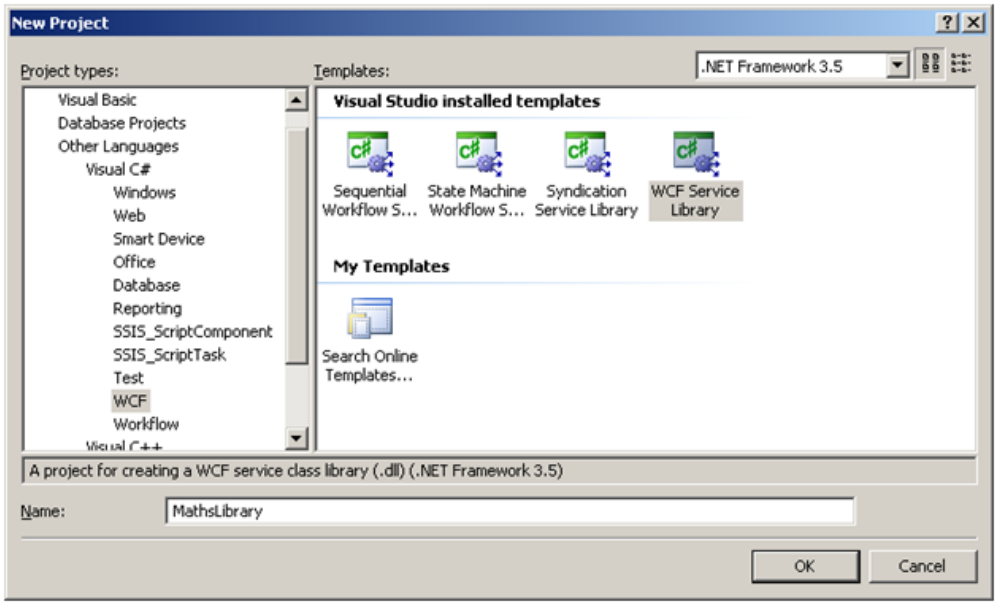
Applications Réparties
Services WCF (Windows Communication Foundation)

1 Création de services WCF avec C

1.1 Création de Service WCF de base en C #. Net

WCF (Windows Communication Foundation) est une partie de. NET. Nous utiliserons Visual Studio 2012 ou une version postérieure pour ce TD.

Le projet à créer est un New Project -> WCF -> WCFClassLibrary projet comme indiqué ci-dessous



Donner le nom **MathsLibrary** à votre projet.

Le projet contient un échantillon des fichiers Service1.cs & IService.cs.

Nous allons les effacer pour ajouter notre propre service.

Faites un clic droit sur MathsLibrary -> Ajouter -> New Item -> Select Class1.cs.

Renommez-le MathsOperations.cs

Cela va créer un fichier de classe simple. Ouvrez le fichier et rendez la classe publique.

De la même façon ajouter une classe IMathsOperations.cs qui sera une interface, qui fournira une liste de toutes les opérations que le service WCF propose.

Ouvrez IMathsOperations.cs et changer pour **public IMathsOperations**

Ajoutez le code ci-dessous.

Applications Réparties
Services WCF (Windows Communication Foundation)

```
namespace MathLibrary
{
    public interface IMathsOperations
    {

       int Add(int num1, int num2);
       int Multiply(int num1, int num2);
    }
}
```

Pour que IMathsOperations devienne un contrat de service WCF, ajouter un attribut **[ServiceContract]**.

Ainsi toute opération que vous souhaitez rendre visible pour le client doit être décoré avec l'attribut **[OperationContract].**

[ServiceContract] et **[OperationContract]** sont inclus dans l'espace de nom System.ServiceModel

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace MathLibrary
{
    [ServiceContract]
    public interface IMathsOperations
    {
        [OperationContract]
        int Add(int num1, int num2);

        [OperationContract]
        int Multiply(int num1, int num2);
    }
}
```

Maintenant, une fois le contrat fixé, nous pouvons implémenter cette interface dans notre service comme ci-dessous

Applications Réparties
Services WCF (Windows Communication Foundation)

```
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace MathLibrary
{
    public class MathsOperations : IMathsOperations
    {
        public int Add(int num1, int num2)
        {
            return(num1 + num2);
        }

        public int Multiply(int num1, int num2)
        {
            return (num1*num2);
        }

    }
}
```

Générez le projet une fois que vous avez terminé à cela.

Ajoutez ou modifier App.config dans la solution. App.config contient des détails sur les endpoints, ce qui comprend les paramètres ABC (Address et Binding principalement).

- Address est l'adresse Où le service peut être trouvé.
- Binding est le Comment accéder aux services
- Contrat est ce que (Quoi) contient le service contient.

Maintenant, nous allons modifier App.config

Si vous faites un clic droit sur App.config, vous pouvez cliquer sur Modifier la configuration WCF.

Vous avez alors accès à une interface graphique qui permet de gérer tous les paramètres du fichier de configuration. Nous vous proposons une série de manipulation au travers cette interface.

Pour une meilleure compréhension des paramètres mieux vaut travailler à même le fichier XML App. Config.

Vous trouverez tous les détails dans http://msdn.microsoft.com/fr-fr/library/ms733932%28v=vs.110%29.aspx.

1.1.1 Sections majeures du fichier de configuration

Les sections principales dans le fichier de configuration incluent les éléments suivants.

Applications Réparties
Services WCF (Windows Communication Foundation)

```
<system.ServiceModel>

  <services>
  <!—- Define the service endpoints. This section is optional in the new
   default configuration model in .NET Framework 4. -->
    <service>
            <endpoint/>
    </service>
  </services>

  <bindings>
  <!-- Specify one or more of the system-provided binding elements,
   for example, <basicHttpBinding> -->
  <!-- Alternatively, <customBinding> elements. -->
    <binding>
    <!-- For example, a <BasicHttpBinding> element. -->
    </binding>
  </bindings>

  <behaviors>
  <!-- One or more of the system-provided or custom behavior elements. -->
    <behavior>
    <!-- For example, a <throttling> element. -->
    </behavior>
  </behaviors>

</system.ServiceModel>
```

Remarque :

Les sections de liaison et de comportement sont facultatives et sont incluses uniquement si besoin est.

- **L'élément <services>**

  L'élément services contient les caractéristiques pour tous les services que l'application héberge. Depuis l'apparition du modèle de configuration simplifié dans .NET Framework 4, cette section est facultative.

  <services>

- **L'élément <service>**

  Chaque service a les attributs suivants :

Applications Réparties
Services WCF (Windows Communication Foundation)

- ✓ name .Spécifie le type qui fournit une implémentation d'un contrat de service. Il s'agit d'un nom complet composé de l'espace de noms, d'un point et du nom du type. Par exemple, "MyNameSpace.myServiceType".
- ✓ behaviorConfiguration .Spécifie le nom de l'un des éléments behavior recherché dans l'élément behaviors. Le comportement spécifié gouverne des actions comme l'autorisation de l'emprunt d'identité par le service. Si sa valeur correspond au nom vide ou qu'aucun behaviorConfiguration n'est fourni, l'ensemble de comportements de service par défaut est ajouté au service.

- *<service>*

- **L'élément <endpoint>**

  Chaque point de terminaison requiert une adresse, une liaison et un contrat, représentés par les attributs suivants :

✓ address .Spécifie l'URI (Uniform Resource Identifier) du service, qui peut être une adresse absolue ou une adresse donnée relativement à l'adresse de base du service. Si l'attribut a une valeur de chaîne vide, il indique que le point de terminaison est disponible à l'adresse de base spécifiée lors de la création de ServiceHost pour le service.

✓ binding .En général, spécifie une liaison fournie par le système comme WsHttpBinding, mais peut également spécifier une liaison définie par l'utilisateur. La liaison spécifiée détermine le type de transport, de sécurité et d'encodage utilisé, et si des sessions fiables, des transactions ou la diffusion en continu sont pris en charge ou activés.

✓ bindingConfiguration .Si les valeurs par défaut d'une liaison doivent être modifiées, cela peut être fait en configurant l'élément binding approprié dans l'élément bindings. Cet attribut doit avoir la même valeur que l'attribut name de l'élément binding utilisé pour modifier les valeurs par défaut. Si aucun nom n'est donné ou qu'aucun bindingConfiguration n'est spécifié dans la liaison, la liaison par défaut du type de liaison est utilisée dans le point de terminaison.

✓ contract .Spécifie l'interface qui définit le contrat. C'est l'interface implémentée dans le type Common Language Runtime (CLR) spécifié par l'attribut name de l'élément service.

<endpoint> element reference

- **L'élément <bindings>**

  L'élément bindings contient les caractéristiques pour toutes les liaisons qui peuvent être utilisées par tout point de terminaison défini dans un service.

  <bindings>

- **L'élément <binding>**

  Les éléments binding contenus dans l'élément bindings peuvent être une des liaisons fournies par le système (consultez Liaisons fournies par le système) ou une liaison personnalisée (consultez Liaisons personnalisées). L'élément binding a un attribut name qui correspond la liaison avec le point de

terminaison spécifié dans l'attribut bindingConfiguration de l'élément endpoint. Si aucun nom n'est spécifié, cette liaison correspond à la valeur par défaut de ce type de liaison.

<binding>

- **L'élément <behaviors>**

  C'est un élément conteneur des éléments behavior qui définissent les comportements pour un service.

  <behaviors>

- **L'élément <behavior>**

  Chaque élément behavior est identifié par un attribut name et fournit un comportement fourni par le système, tel que <throttling>, ou un comportement personnalisé. Si aucun nom n'est donné, cet élément behavior correspond au service par défaut ou au comportement du point de terminaison.

  <behavior> de <serviceBehaviors>

Configurez votre fichier App.Config pour créer un service utilisant la bibliothèque MathsLibrary.dll, et offrant un endpoint.

La configuration du endpoint définira son nom, le contrat qui lui correspond, le binding BasicHttpBinding et le "Host" qui correspond à son adresse.

1.1.2 Génération du projet et lancement du service

Une fois que cela est fait, générez le projet.

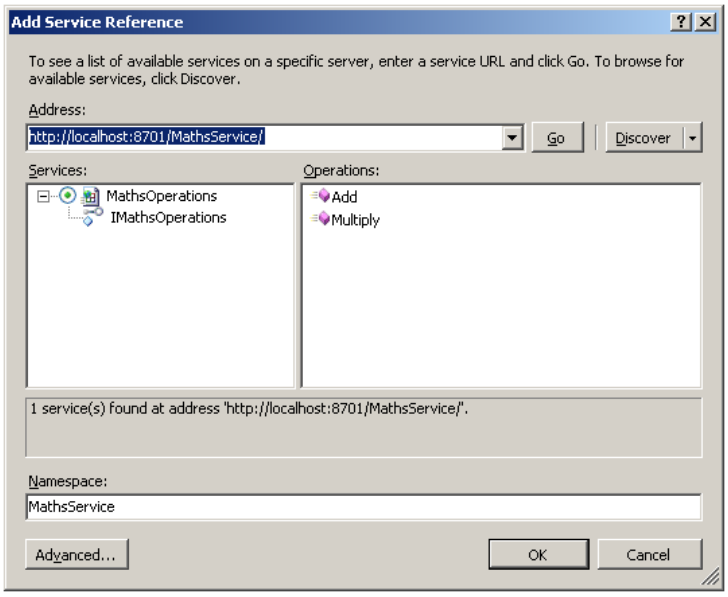
Visual Studio fournit son propre processus d'accueil d'hébergement de services. Nous pouvons également héberger le service comme une application console ou un Service Windows, ou encore dans IIS.

2 Création d'un client WCF avec C#

Maintenant, nous allons créer une application cliente. Ajouter un projet "application console" (ou tout autre type de projet) dans la même solution ou une solution différente.

Ajouter une référence au service MathsService que nous avons créé. Il chargera automatiquement toutes les DLL nécessaires.

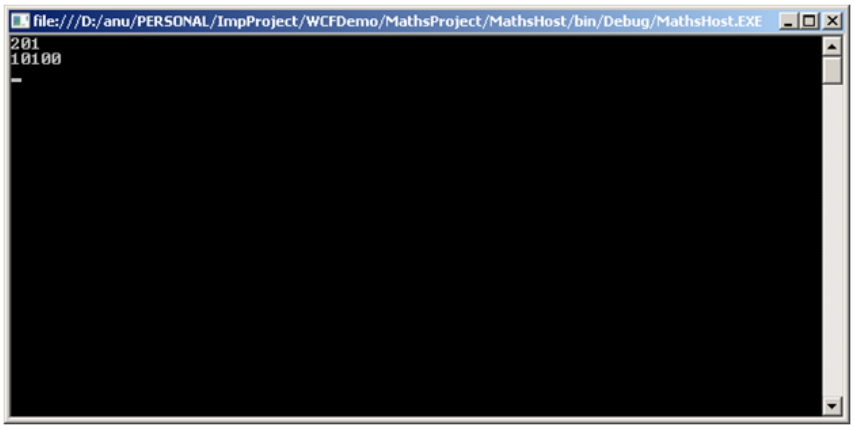
Applications Réparties
Services WCF (Windows Communication Foundation)



Puis, avec le code ci-dessous, nous pouvons appeler le service :

```
class Program
{
    static void Main(string[] args)
    {
        MathsOperationsClient client = MathsOperationsClient();
        Console.WriteLine(client.Add(100, 101));
        Console.WriteLine(client.Multiply(100, 101));
        Console.ReadLine();
    }
}
```

Cela vous donnera dessous de la sortie



Applications Réparties
Services WCF (Windows Communication Foundation)

3 Exploration de différents Bindings

Sans modifier vos codes sources, modifiez et testez de nouveaux bindings entre le client et le serveur.

Créer pour cela plusieurs endpoints avec des bindings différents et testez-les.

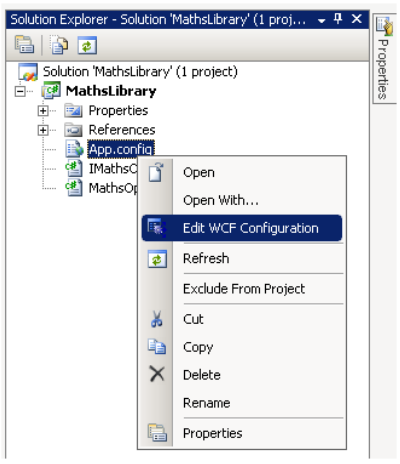
Attention dans ce cas de passer le nom du endpoint que vous voulez adresser lors d'une instanciation de la classe proxy sur le service. C'est l'un des noms du endpoints locaux au client qui ont été générés lors de la création de la classe du proxy et qui figure dans le fichier App.Config du client (sinon il faudra le changer à la main …).

Exemple: MathsOperationsClient client = new MathsOperationsClient("BasicHTTP");

Dans le cas:

```
<endpoint address="" binding="basicHttpBinding" name="BasicHTTP",
contract="MathsLibrary.IMathsOperations">
       <identity>
              <dns value="localhost" />
       </identity>
</endpoint>
```

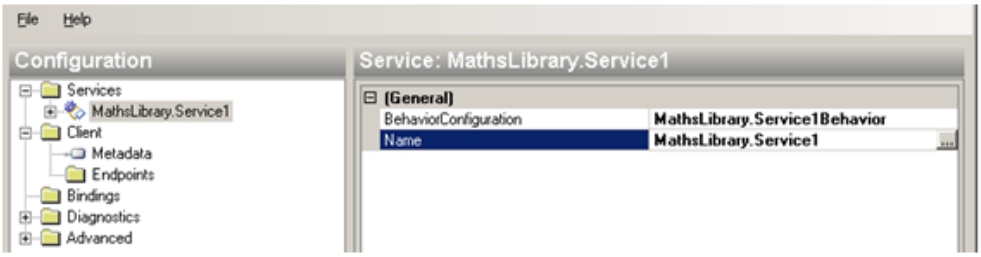
4 Annexe : Interface graphique d'édition des paramètres de configuration d'App.Config



Le popup ci-dessous doit apparaître

Applications Réparties
Services WCF (Windows Communication Foundation)

Sélectionnez le Service1 du panneau de gauche, la fenêtre suivante doit apparaître

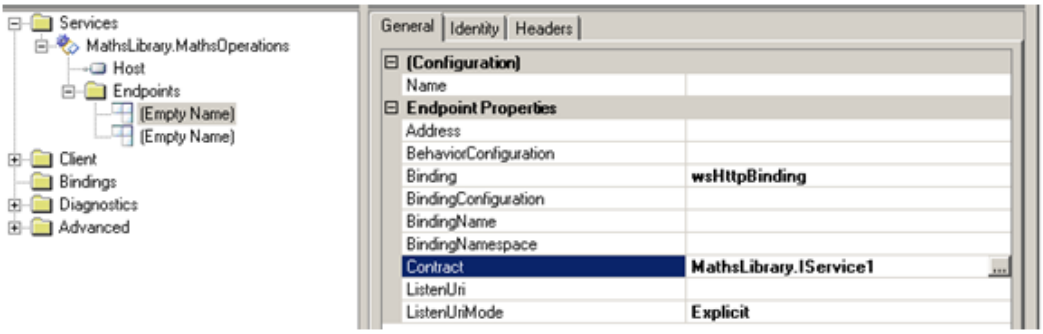


Cliquez sur le bouton "points de suspension" et trouvez MathsLibrary.dll.

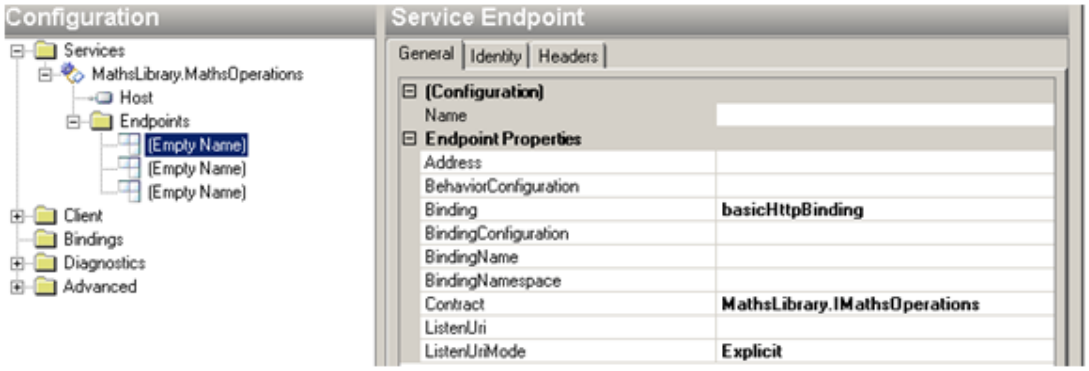
Cliquez sur ce lien, il vous donnera le nom du service qu'elle contient.

De même aller sur des endpoints, et sélectionnez un contrat approprié avec les mêmes étapes que ci-dessus.

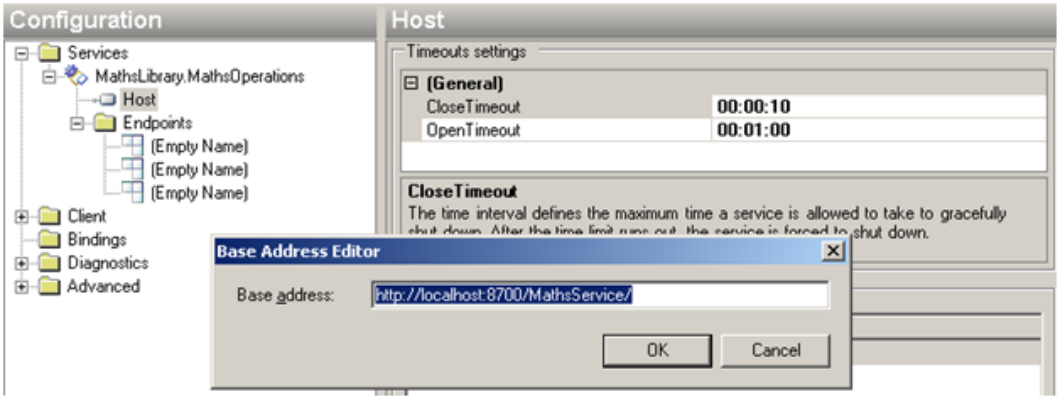
Applications Réparties
Services WCF (Windows Communication Foundation)



Nous pouvons aussi changer le binding comme ci-dessous ...



Sélectionnez "Host" sur le côté gauche et vous verrez l'adresse de base. Vous pouvez la modifier vers l'adresse que vous voulez.



Une fois que cela est fait, générez le projet.