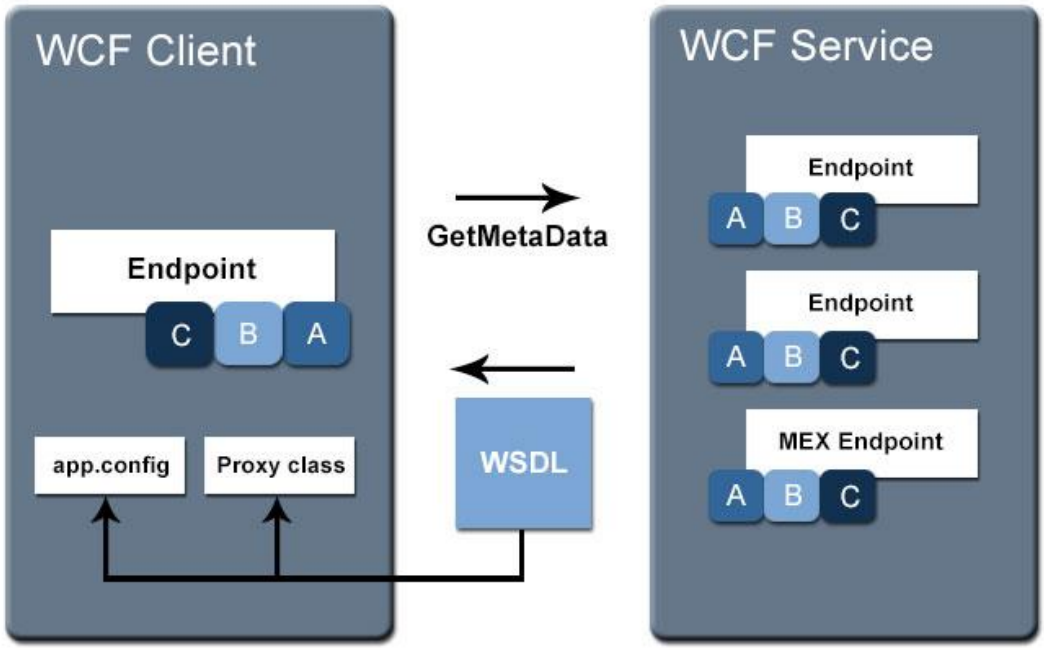
Applications Réparties
Services WCF et Metadata Exchange

1 Introduction à l'échange de MetaDonnées ou Metadata Exchange

Les Métadonnées dans WCF se réfèrent aux informations qui décrivent précisément comment communiquer avec un service. Les clients peuvent demander les métadonnées à partir d'un service en cours pour connaître ses points de terminaison et les formats de message utilisés. Au moment de la **conception**, les clients envoient un message de demande défini par la norme WS-MetadataExchange et reçoivent des données au format WSDL en retour. Le WSDL est alors utilisé par le client pour définir et généré un fichier de classe de proxy et la configuration qui sera ensuite utilisé lors de l'exécution pour communiquer avec le service.



Par défaut, les services WCF expose un point de terminaison MEX. Cela signifie que le client peut interroger le service pour savoir comment communiquer avec lui. Nous avons déjà utilisé ce point de terminaison sans le savoir puisque l'IDE lance un client généré grâce aux Metadonnées pour tester et debugger un service en cours de développement (Cf. les TDs précédents).

2 Comment : publier les métadonnées d'un service à l'aide d'un fichier de configuration

Il y a deux façons de spécifier comment un service doit publier des métadonnées : à l'aide d'un fichier de configuration et à l'aide du code. Cette rubrique montre comment publier des métadonnées pour un service à l'aide d'un fichier de configuration.

Cette rubrique indique comment publier des métadonnées de manière non sécurisée. Tout client peut récupérer

Applications Réparties
Services WCF et Metadata Exchange

les métadonnées du service. Si votre service doit publier les métadonnées de manière sécurisée, il vous faudra mettre en place un point de terminaison de métadonnées sécurisé personnalisée.

La publication des métadonnées permet aux clients de récupérer les métadonnées via une requête WS-Transfer GET ou une requête HTTP/GET à l'aide de la chaîne de requête **?wsdl**. Pour être sûr que le code fonctionne, créez un service WCF de base. Pour plus de simplicité, un service auto-hébergé de base est fourni dans le code suivant.

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

namespace Metadata.Samples
{
   [ServiceContract]
   public interface ISimpleService
   {
      [OperationContract]
      string SimpleMethod(string msg);
   }

   class SimpleService : ISimpleService
   {
      public string SimpleMethod(string msg)
      {
         Console.WriteLine("The caller passed in " + msg);
         return "Hello " + msg;
      }
   }

}
```

Le fichier de configuration suivant sert de point de départ. N'oubliez pas de rajouter l'adresse de base (base address)

```
<configuration>
 <system.serviceModel>
  <services>
   <service name="Metadata.Example.SimpleService">
    <endpoint address=""
            binding="basicHttpBinding"
            contract="Metadata.Example.ISimpleService" />
   </service>
  </services>
  <behaviors>

  </behaviors>
 </system.serviceModel>
</configuration>
```

Applications Réparties
Services WCF et Metadata Exchange

2.1 Pour publier les métadonnées d'un service WCF à l'aide d'un fichier de configuration d'application

1. Dans le fichier App.config, après la fermeture de l'élément **</services>**, créez un élément **<behaviors>**.
2. Dans l'élément **<behaviors>**, ajoutez un nouvel élément **<serviceBehaviors>**.
3. Ajoutez un élément **<behavior>** à l'élément **<serviceBehaviors>** et spécifiez une valeur pour l'attribut **name** de l'élément **<behavior>**.
4. Ajoutez un élément **<serviceMetadata>** à l'élément **<behavior>**.Affectez à l'attribut **httpGetEnabled** la valeur **true** et à l'attribut **policyVersion** la valeur Policy15.

   httpGetEnabled permet au service de répondre aux demandes de métadonnées faites par une demande HTTP GET.

   policyVersion indique au service de se conformer à WS-Policy 1.5 lors de la génération des métadonnées.

5. Ajoutez un attribut **behaviorConfiguration** à l'élément **<service>** et spécifiez l'attribut **name** de l'élément **<behavior>** ajouté à l'étape 1, comme illustré dans l'exemple de code suivant.

```
<services>
 <service
    name="Metadata.Example.SimpleService"
    behaviorConfiguration="SimpleServiceBehavior">
  ...
 </service>
</services>
<behaviors>
 <serviceBehaviors>
  <behavior name="SimpleServiceBehavior">
   <serviceMetadata httpGetEnabled="True" policyVersion="Policy15" />
  </behavior>
 </serviceBehaviors>
</behaviors>
```

6. Ajoutez un ou plusieurs éléments **<endpoint>** et attribuez au contrat la valeur **IMetadataExchange**, comme illustré dans l'exemple de code suivant.

```
<services>
 <service
    name="Metadata.Example.SimpleService"
    behaviorConfiguration="SimpleServiceBehavior">

  <endpoint address=""
        binding="wsHttpBinding"
        contract="Metadata.Example.ISimpleService" />
```

Applications Réparties
Services WCF et Metadata Exchange

```
  <endpoint address="mex"
        binding="mexHttpBinding"
        contract="IMetadataExchange" />
 </service>
</services>
```

7. Pour les points de terminaison de métadonnées ajoutés à l'étape précédente, affectez à l'attribut **binding** l'une des valeurs suivantes :
   - **mexHttpBinding** pour la publication HTTP.
   - **mexHttpsBinding** pour la publication HTTPS.
   - **mexNamedPipeBinding** pour la publication de canal nommé.
   - **mexTcpBinding** pour la publication TCP.
8. Pour les points de terminaison de métadonnées ajoutés à l'étape précédente, attribuez à l'adresse la valeur suivante :
   - Une chaîne vide pour utiliser l'adresse de base de l'application hôte comme point de publication si l'adresse de base est la même que la liaison de métadonnées.
   - Une adresse relative si l'application hôte a une adresse de base.
   - Une adresse absolue.
9. Créez et exécutez l'application console.
10. Utilisez Internet Explorer pour naviguer jusqu'à l'adresse de base du service (http://localhost:8001/MetadataSample dans cet exemple) et vérifiez que la publication des métadonnées est activée. Dans la négative, un message s'affiche en haut de la page résultante : "La publication des métadonnées pour ce service est actuellement désactivée".

2.2 Pour utiliser les points de terminaison par défaut

Pour configurer des métadonnées sur un service qui utilise les points de terminaison par défaut, spécifiez ServiceMetadataBehavior dans le fichier de configuration, comme dans l'exemple précédent, mais ne spécifiez aucun point de terminaison. Le fichier de configuration se présenterait alors comme suit.

```
<configuration>
 <system.serviceModel>
  <behaviors>
   <serviceBehaviors>
    <behavior name="SimpleServiceBehavior">
     <serviceMetadata httpGetEnabled="True" policyVersion="Policy12" />
    </behavior>
   </serviceBehaviors>
  </behaviors>

 </system.serviceModel>
</configuration>
```

Étant donné que le service a un ServiceMetadataBehavior avec le **httpGetEnabled** ayant la valeur **true**, la publication des métadonnées est activée pour le service, et comme aucun point de terminaison n'a été ajouté explicitement, le runtime ajoute les points de terminaison par défaut. Pour plus d'informations sur les points de terminaison, les liaisons et les comportements par défaut, consultez Configuration simplifiée et Configuration simplifiée pour WCF Services.

Applications Réparties
Services WCF et Metadata Exchange

L'exemple de code suivant affiche l'implémentation d'un service WCF de base et le fichier de configuration qui publie les métadonnées pour le service.

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

namespace Metadata.Samples
{
   [ServiceContract]
   public interface ISimpleService
   {
      [OperationContract]
      string SimpleMethod(string msg);
   }

   class SimpleService : ISimpleService
   {
      public string SimpleMethod(string msg)
      {
         Console.WriteLine("The caller passed in " + msg);
         return "Hello " + msg;
      }
   }

   class Program
   {
      static void Main(string[] args)
      {
         ServiceHost host = new ServiceHost(typeof(SimpleService),
            new Uri("http://localhost:8001/MetadataSample"));
         try
         {
            // Open the service host to accept incoming calls
            host.Open();

            // The service can now be accessed.
            Console.WriteLine("The service is ready.");
            Console.WriteLine("Press <ENTER> to terminate service.");
            Console.WriteLine();
            Console.ReadLine();

            // Close the ServiceHostBase to shutdown the service.
            host.Close();
         }
         catch (CommunicationException commProblem)
         {
            Console.WriteLine("There was a communication problem. " + commProblem.Message);
```

Applications Réparties
Services WCF et Metadata Exchange

```
            Console.Read();
        }
    }
  }
}
```

```
<configuration>
  <system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior name="SimpleServiceBehavior">
          <serviceMetadata httpGetEnabled="True" policyVersion="Policy12" />
          <serviceDebug includeExceptionDetailInFaults="False" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
</configuration>
```

3 Rappel : pour démarrer à la fois le client et l'hôte à partir de Visual Studio

- Créez une solution Visual Studio qui contient à la fois les projets client et serveur.
- Configurez la solution de façon à ce qu'elle démarre à la fois les processus client et serveur lorsque vous choisissez **Démarrer** dans le menu **Déboguer**.
    - Dans l'**Explorateur de solutions**, cliquez avec le bouton droit sur le nom de la solution.
    - Cliquez sur **Définir les projets de démarrage**.
    - Dans la boîte de dialogue **Propriétés de la Solution <Nom>**, sélectionnez **Plusieurs projets de démarrage**.
    - Dans la grille **Plusieurs projets de démarrage**, sur la ligne qui correspond au projet serveur, cliquez sur **Action** et choisissez **Démarrer**.
    - Sur la ligne qui correspond au projet client, cliquez sur **Action** et choisissez **Démarrer**.
    - Cliquez sur **OK**.