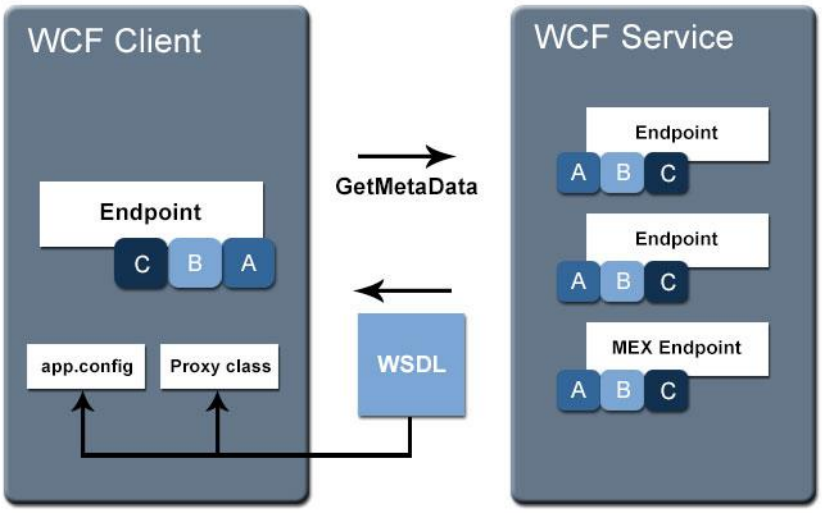
Applications Réparties
Services WCF et Metadata Exchange

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.

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**.

QUESTION 1 : Pour être sûr que le code fonctionne, créez un service WCF de base avec un projet du type « Bibliothèques de service WCF ». Démarrez ce service et vérifiez son fonctionnement à partir du client de debugging de l'IDE.

QUESTION 2 : Vérifiez la disponibilité du service via son adresse Web (ex. http://localhost:8733/Design_Time_Addresses/WcfServiceLibrary1/Service1/). En invoquant l'url http://localhost:8733/Design_Time_Addresses/WcfServiceLibrary1/Service1/?wsdl , que récupérez-vous ?

QUESTION 3 : Créez et modifiez un nouveau projet WCF / Bibliothèque de Service WCF dans lequel vous insérerez tout ou partie du code suivant pour implémenter dans un espace de nom Metadata.Samples

Applications Réparties
Services WCF et Metadata Exchange

un service SimpleService avec une méthode SimpleMethod qui retournera un message de type String contenant "Hello " suivi d'une chaine de caractères saisie au clavier côté serveur.

```
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);
   }

}
```

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

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

QUESTION 4 : Modifiez le fichier App.config en conséquence et testez-le comme le projet précédent. A chaque modification noté dans votre fichier, en commentaires, les raisons de vos modifications. Vous pouvez utiliser l'outil graphique pour les modifications du fichier de configuration et vous référez aux explications du TD précédent.

QUESTION 5 : Créer dans une nouvelle instance de Visual Studio un projet « Application Console ». Ajouter une référence Web sur le projet Metadata.Samples après avoir lancé ce dernier. Ecrivez quelques lignes de code pour tester le service depuis le client.

Applications Réparties
Services WCF et Metadata Exchange

3 Consommation de services facilitée avec un contrat fort de type WSDL

Lors du TD 2 de ce module, nous avons vu comment accéder à un service Web de type http/REST pour obtenir des informations temps réel sur la disponibilité des Vélibs de Paris.

Pour rappel, le service permet de visualiser les disponibilités des vélos et des emplacements de parking à la disposition des parisiens dans le cadre du service Vélib géré par la ville de Paris :

a) La liste des points d'accès Vélib et leurs localisations

b) La dispo des Vélibs en temps réel à chaque point d'accès

Les URI de la forme http://www.velib.paris/service/stationdetails/<number> fournissent les disponibilités des points d'accès. Sur chaque point d'accès <number>, le champ <available> donne le nombre de vélos disponibles, le champ <free> le nombre d'emplacements libres, le champ <total> donne le nombre d'emplacements, le champ <ticket> à 1 si le paiement par carte bleue est accepté. Les trois champs restants sont non documentés.

Malheureusement ce type de service est utilisable au travers un **contrat** du « **faible** » qui ne permet pas la génération automatique de classes proxy côté client.

QUESTION 6 : Nous allons dans cette partie créer un service de type WS-SOAP qui donnera accès au service http://www.velib.paris/service/stationdetails/<number> avec une API SOAP et surtout qui fournira un **contrat** « **fort** » au travers une description **WSDL**. Tester ce nouveau service avec un client dans une autre session de Visual Studio est en intégrant une référence web à partir du wsdl exposé par votre service.

QUESTION 7 : Reprenez l"exercice précédent en rajoutant votre propre Service et API pour accéder à votre code développé dans le TD2 section 2.3 question 3. Pour rappel, il s'agissait d'écrire un client qui vous donne le nombre de Vélibs disponibles à un point d'accès dont vous indiquiez le nom. Le point d'accès était identifié grâce à une sous-chaine présente dans son nom <name>. Vous en faites ainsi un