

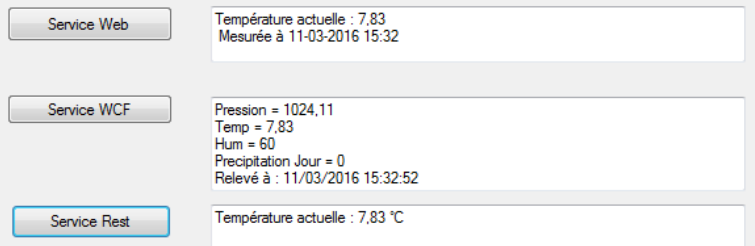
TP .Net n° 2 : Web Services, servives WCF, ADO.Net

Objectif : Utilisation et création de Web Services et de servive WCF avec différents types d'applicatifs

Partie 1 : Utilisation de services dans une application

Lancez « Visual Studio » et créez un nouveau projet de type « C# - Windows Forms (.Net Framework) ».

Ajoutez graphiquement 3 boutons et et 3 contrôles « TextBox » multilines comme sur la figure ci-contre. Interceptez les fonctions C# liées au click des boutons (double-clic sur ceux-ci).



Dans un navigateur tapez l'Url : <http://www.meteoservice.lab3il.fr/meteo3il.asmx>

A quoi correspond cette Url ? : _____

A quelle url se situe le contrat WSDL ? : _____

Sous quelle forme se présente t-il ? : _____

Ce service Web est un service de météorologie permettant d'obtenir les valeurs mesurées par la station météo située sur le toit du bâtiment de 3iL (description sur <http://www.meteo.lab3il.fr/Service.aspx>).

Dans votre projet, faites « Projet/Ajouter une référence de service », et dans l'onglet « URL » tapez l'url du service et cliquez sur le bouton « Aller à ». Que s'affiche t-il (sélectionnez notamment *meteo3ilSoap*) ? :

Renommez votre espace de noms en « WSMeteo » et faites « OK ». Dans l'explorateur de votre projet, à quel endroit votre référence de service apparaît-elle ? :

Pour utiliser le service Web, vous devez instancier un objet « service » de la classe « *meteo3ilSoapClient* » avec l'opérateur « new » qui représente le service qui va nous donner accès aux différentes méthodes.

```
WSMeteo.meteo3ilSoapClient service = new WSMeteo.meteo3ilSoapClient("meteo3ilSoap");
```

Un service Web peut être utiliser en mode synchrone (appel direct des méthodes) ou en mode asynchrone (utilisation pour chaque méthode d'un événement qui va se déclencher à la réception de la réponse du service, souvent utilisé en programmation mobile). Nous allons utiliser le mode synchrone.

Dans le code du premier bouton, instanciez votre service, créez 2 chaînes de caractères (Date et Desc) et appelez la méthode « Get_Value » du service (« *service.Get_Value()* »), en passant en paramètre le nombre 1 (pour la température) et vos 2 chaînes de caractères précédées de « out » car celles sont des variables retournées

par la méthode. Affichez le résultat dans le « TextBox » associé. Testez votre application pour vérifier que la valeur est bien renvoyée.

Ajoutez comme nouvelle référence nommée « WCFMeteo » (« Projet/Ajouter une référence de Service »), le deuxième service fournit par la station situé à l'adresse : http://www.meteowcfservice.lab3il.fr/meteo3il_2.svc

De quel type de service s'agit-il ? : _____

Le deuxième bouton va afficher les données météo de ce service. Vous allez instancier le service avec la syntaxe :

```
WCFMeteo.Service1Client sr = new WCFMeteo.Service1Client();
```

Ensuite vous devez appeler sa méthode « `GetMeteoData` » de façon synchrone. Cette fonction retourne un objet « `MeteoData` » contenant en propriétés les valeurs des différents champs. Vous devez créer un objet de la classe « `WCFMeteo.MeteoData` » et qui va être chargé par l'appel à cette méthode, puis lire ses propriétés (d_Temp, d_Pres, i_Hum, ...)

La méthode et l'objet retourné sont décrits sur : <http://www.meteo.lab3il.fr/Service.aspx>

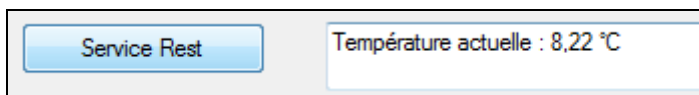
Le dernier format de service Web à tester est le format Rest. Les service Rest ne sont pas propres à la technologie .Net et ne sont donc pas importés en tant que référence de service.

- Ajoutez les 2 namespaces `System.Net` et `System.IO`.
- Créez une chaîne contenant l'Url : <http://www.meteorestservice.lab3il.fr/ServiceRest.svc/meteo/2>
- Créez un objet de type `HttpWebRequest` qui vaut : `(HttpWebRequest)WebRequest.Create(url)`
- Appelez la propriété `Method` de cet objet pour lui préciser que c'est appel de type Get `(WebRequestMethods.Http.Get)`
- Récupérez un objet de type `WebResponse` en appelant la méthode `GetResponse` de l'objet `HttpWebRequest`
- Créez un objet de type `StreamReader` (avec `new`) en appelant dans son constructeur la méthode `GetResponseStream` de l'objet `WebResponse`

Vous pouvez afficher directement le contenu du `StreamReader` en appelant la méthode `ReadLine`. Cependant la réponse est encapsulée dans du code qui n'est ni du Xml ni du Json. Vous pouvez vous débarrasser de ce code en utilisant :

```
str = str.Substring(str.IndexOf(">") + 1, str.Length - str.IndexOf(">") - 1);  
str = str.Substring(0, str.IndexOf("<"));
```

A l'arrivée, vous devez afficher :



Partie 2 : Création et utilisation d'un service WCF

Démarrez une nouvelle instance de Visual Studio, et faites menu « C# - Service WCF » (Si Visual Studio ne vous propose pas ce type de projet, démarrez l'installateur, faites « Modifier/Composants Individuels », tapez « WCF » et installez la Package).

Affichez le code de votre service. Quel est l'espace de noms spécifique aux services Web ? :

Lancez un test de votre service en sélectionnant un navigateur pour cible. Lorsque le navigateur est affiché, sélectionnez le fichier « Service1.svc ». Que génère Visual Studio et quelle application est utilisée pour le test ? :

Quelle est l'url correspondant à votre service ? : _____

Quelle est l'extension de votre service ? : _____

A quelle url se trouve le contrat WSDL ? : _____

Arrêtez votre service et ouvrez le code. Quel est le mot clé permettant de spécifier qu'une méthode est callable à travers le service Web WCF ? : _____

En vous appuyant sur le code exemple de la méthode « GetData » :

La déclaration d'une méthode se situe dans le fichier : _____

Le code d'une méthode se situe dans le fichier : _____

Maintenant créez une méthode nommée « Meteo_GetTemperature » qui va dans un premier temps renvoyer la dernière température relevée par la station sous forme de chaîne de caractères.

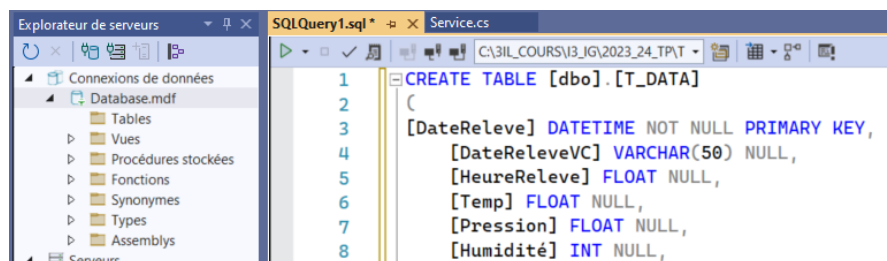
A - PC de l'école ou portable connecté en ethernet sur le réseau de 3iL :

Les données de température se situent dans la base de données « StationMeteo » située sur « srv-sql ». Vous pouvez y accéder en utilisant le compte « UserMeteo » avec le mot de passe « MeteoUser ».

Quelle chaîne de connexion devez-vous utiliser ? : _____

B. - Portable non connecté au réseau de l'école

Sur votre projet, faites Ajouter/Nouvel Elément/Base de données SQL-Server. Sur « Database.mdf » avec le bouton droit de la souris faites « Ouvrir », puis dans l'explorateur de base de données faites avec le bouton droit de la souris « Nouvelle Requête », copiez y le script SQL « TP2_Database.txt » qui vous est fourni et exécutez-le.



Votre base de données locale est prête. Vous devez utiliser la chaîne de connexion suivante :

@ "Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\.....\Database1.mdf;Integrated Security=True;"

A ou B étant réalisé, reprenez.

Vous devez programmer cette méthode en accédant à la base de données via ADO.NET et en utilisant des objets « SqlConnection », « SqlCommand » et « SqlDataReader » (voir cours). Pensez à inclure l'espace de nom « System.Data.SqlClient ». La requête SQL pour lire le dernier jeu de valeurs mesurées est :

```
SELECT * FROM [T_Data] WHERE [DateReleve] = (SELECT MAX([DateReleve]) FROM [T_Data])
```

Vous pouvez récupérer la valeur de température lue par le « datareader » dans une variable de type « double » :

```
temp = dr.GetDouble(dr.GetOrdinal("Temp"));
```

Testez votre méthode dans un navigateur et vérifiez qu'elle fonctionne correctement.

Lancez votre service et laissez le tourner dans votre navigateur.

Revenez à votre premier projet pour y tester votre service. Dans ce projet, il faut bien sûr créer une référence web pointant sur votre service en utilisant l'URL local de votre service (affiché dans votre navigateur). Ajoutez

ensuite un bouton et le code qui instancie votre service WCF, appelle votre méthode en synchrone et affiche la température.

(Lorsque vous avez modifié et relancé votre service, pour prendre en compte ces modifications dans votre projet client, cliquez avec le bouton droit de la souris sur la référence à votre service et sélectionnez « Mettre à jour la référence de service »).

Maintenant créez une méthode nommée « `Meteo_GetTemperatureByDate` » qui va renvoyer la température relevée par la station la plus proche de la date passée en paramètre. Cette méthode doit posséder 2 arguments de type string : la date de la mesure demandée et une variable qui recevra la date exacte de la mesure. Pour signaler que le 2° argument est une valeur de retour, il faut le déclarer « out » : « out string variable ». Cette méthode retourne la valeur de température demandée sous forme de chaîne.

La requête Sql à utiliser est sous cette forme :

```
SELECT * FROM [T_Data] WHERE [DateReleve] = (SELECT MAX([DateReleve]) FROM [T_Data]
WHERE DateReleve < '16/03/2015 15:30')
```

La valeur à retourner est le champ « Temp » à récupérer en « Double » comme précédemment. La valeur de la date du relevé lue par le « datareader » est à récupérer dans le champ « `DateReleveVC` » sous forme de chaîne et sera renvoyé dans la variable du 2° argument de la méthode : `dr.GetString(dr.GetOrdinal("DateReleveVC"))`;

Lancez votre service et laissez le tourner dans votre navigateur et revenez à votre projet pour y tester cette méthode. Il faut bien sûr mettre à jour la référence de service et modifier le code pour utiliser cette nouvelle méthode en asynchrone.

Partie 3 : Transmission d'un DataSet par un service WCF

Dans votre service, vous allez créer une nouvelle méthode « `Meteo_GetTemperatureDay` » qui va renvoyer toutes les mesures de température du jour correspond à la date passée en paramètre.

La requête à utiliser est de la forme suivante :

```
SELECT * FROM [T_Data] WHERE [DateReleve] > 'madate' AND [DateReleve] < DateAdd(d, 1, 'madate' )
```

Vous devez créer un objet « `DataAdapter` » pour réaliser la requête et un objet « `DataSet` » qui sera chargé avec les données retournées par la méthode « `Fill` » du « `DataAdapter` » (voir cours).

Qu'est ce qu'un « `Dataset` » ? :

Quel est le rôle la fonction « `Fill` » ? :

N'oubliez pas de renvoyer votre « `DataSet` » en valeur de retour de votre méthode.

Lancez votre service et laissez le tourner dans votre navigateur. Dans votre 1° projet, mettez à jour la référence à votre service. Ajoutez à votre fenêtre un bouton « Afficher graphique », un « `DateTimePicker` » et un graphe de type « `Chart` ».

Dans le code de la méthode « `click` » de votre bouton, instanciez une variable représentant votre service Web local.

```
monServWCF.Service1Client ms = new monServWCF.Service1Client();
```

Créez ensuite une variable « `DataSet` » et affectez lui le « `DataSet` » renvoyé par la méthode « `Meteo_GetTemperatureDay` » de votre service. Pour définir la date passée en paramètre, utilisez la valeur renvoyée par le « `DateTimePicker` » : `ctp1.Value.Date.ToShortDateString()`;

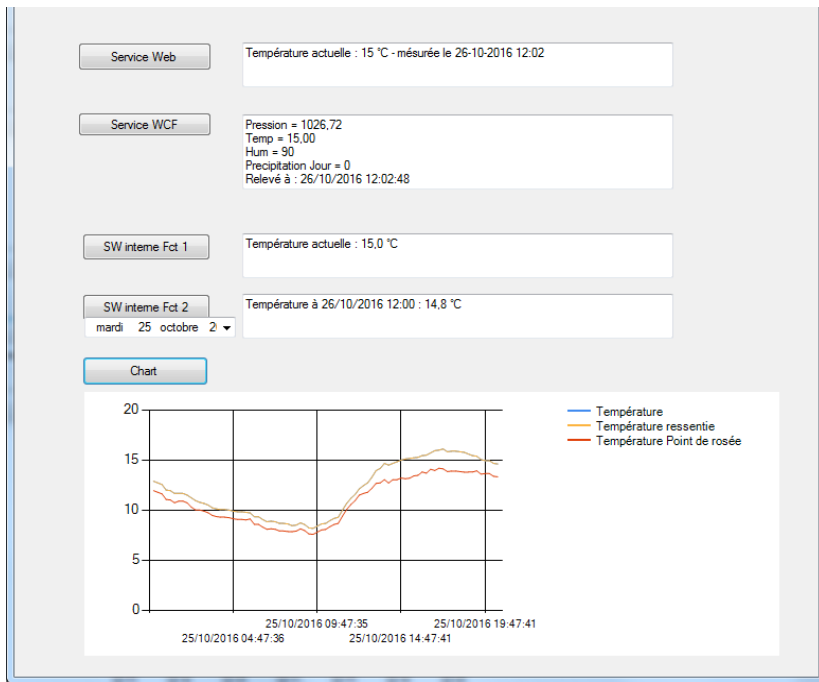
Pour afficher la température dans le contrôle « `Chart` », il faut d'abord lui affecter sa source de donnée par la propriété « `DataSource` » en lui passant une table du « `dataset` » en l'occurrence la seule qu'il possède (indice 0 de la collection « `Tables` » du `dataset`).

Ensuite il faut définir la première série de valeur à afficher :

- Faites un « `chart1.Series.Clear()` » pour tout réinitialiser

- Ajoutez une nouvelle série (propriété Add de Series) en la nommant par exemple « Temp ». Vous pouvez ensuite accéder à toutes ses propriétés par « chart1.Series["Temp"].xxxxx »
- Définissez le champ qui sera utilisé en X par la propriété « XValueMember » ("DateReleveVC")
- Définissez le champ qui sera utilisé en Y ("Temp")
- Définissez le type de tracé à utiliser par la propriété « ChartType »
- Définissez la légende à afficher par la propriété « LegendText »
- Pour finir, appelez la méthode « DataBind » du graphe et mettez sa propriété « Visible » à vrai

Vous avez d'autres données dans votre « DataSet », donc ajoutez 2 autres séries : une qui affiche la température ressentie (« TempR ») et une autre qui affiche la température du point de rosée (« TempPR »).



Quel type d'architecture utilise cette solution applicative ? :

Représentez le schema de cette architecture :