

IG sous DotNet

Ch B - Les accès aux données :
Ado.Net, Entity Framework, Linq
les services Web Soap, WCF, et Rest



B. Chervy

2023

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

Sommaire



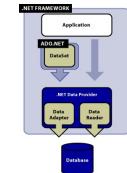
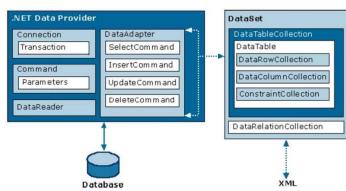
1. - Ado.net
2. - Entity Framework, Linq
3. - Les services Web Soap
4. - Les services WCF
5. - Les Services Rest

Annexes

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL



1. - ADO.Net



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

3

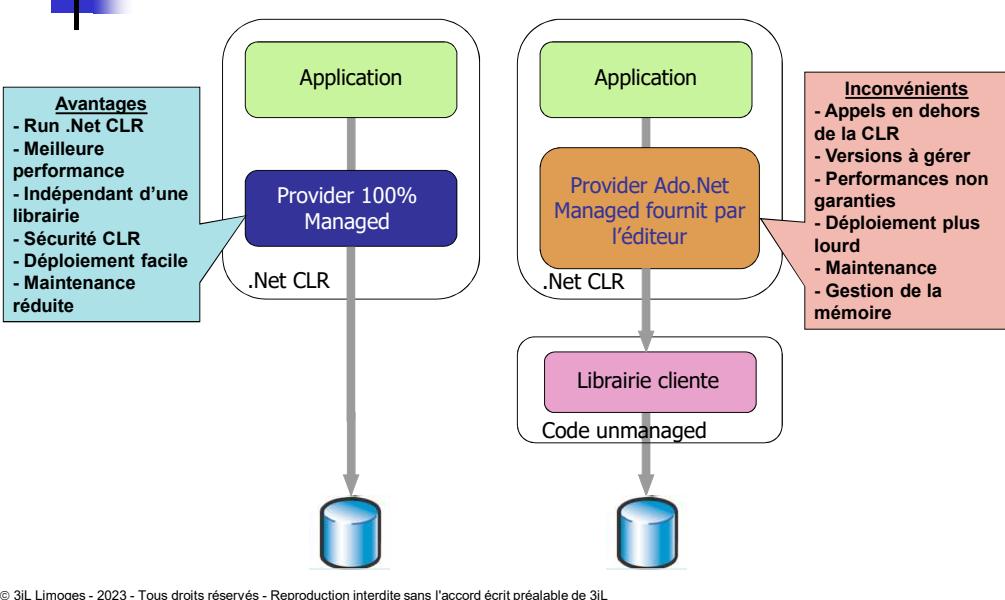
Ado.Net

- **ADO.NET** (ADO pour Activex DataBase Object) :
 - ⇒ **couche d'accès aux bases de données fournie avec le Framework .NET**,
 - ⇒ fonctionne sur le principe de **fournisseurs managés** ou fournisseurs codés
 - ⇒ géré par la plate-forme .NET, tous les objets sont gérés par la CLR
 - ⇒ est entièrement **indépendant** de la base de données (pas les recordset Ado)
 - ⇒ est **interopérable** avec d'autres systèmes
(entièrement conçu avec XML, permet de manipuler des données provenant d'autre chose que de bases de données : ActiveDirectory, Index Server, système de fichiers, ...)
 - ⇒ permet le fonctionnement en **mode connecté** (connexion permanente) ou **mode déconnecté** (connexion libérée, données en mémoire)
- **ADO.NET n'est pas ADO** (recordset utilisant des curseurs) :
 - ⇒ écrit en .Net ⇒ **plus performant** que Ado (composant COM)
 - ⇒ ADO reste disponible pour les programmeurs à travers des services d'interopérabilité COM de .NET (nécessite d'installer MDAC 2.6 ou supérieur pour utiliser les fournisseurs de données natifs comme OLE DB)

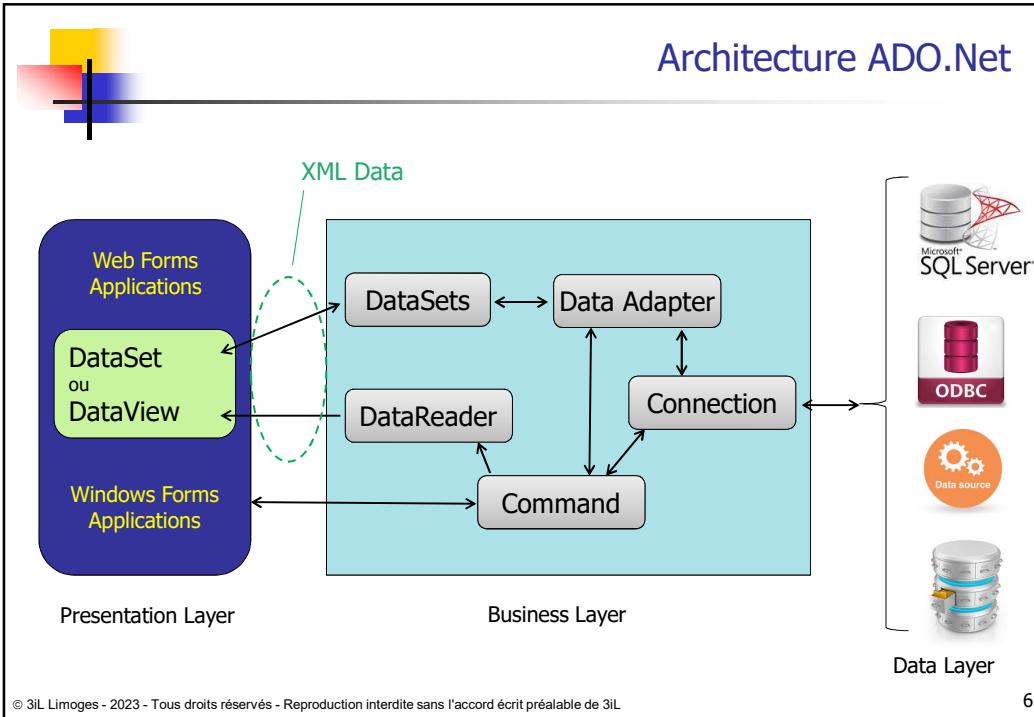
© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

4

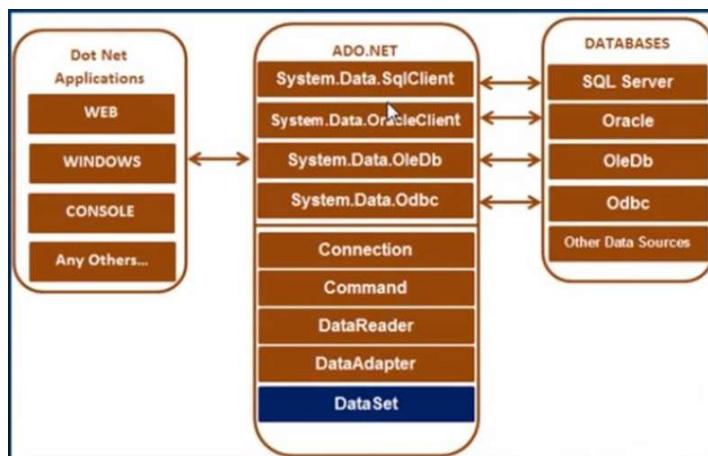
Accès managed et unmanaged



Architecture ADO.NET



Architecture ADO.Net

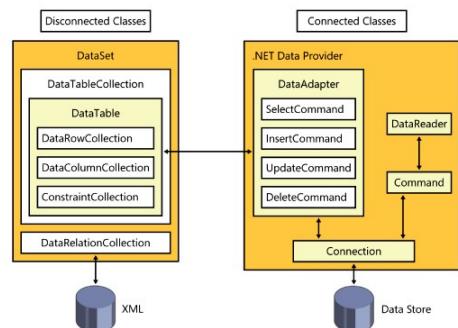


© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

7

Classes et objets

- Les fournisseurs managés permettent l'accès aux sources de données :
⇒ extraient et présentent les données **à travers des classes .NET**
- Comprend les classes : **Command**, **Connection**, **DataReader**, **DataAdapter**, **DataSet** et **Transaction**
⇒ classe préfixée identifiant le fournisseur : **SqlCommand**, **OracleCommand**
- Opérations de base :
 - paramétrage d'une connexion à une source de donnée : objet **Connection**
 - récupération du flux de données en lecture seule : objet **DataReader**.
 - récupération d'un flux d'information, transfert à un objet local pour visualisation et mise à jour : objets de type **DataAdapter** et **DataSet**
 - synchronisation des mises à jour réalisée par rapport à la source de données : objets de type **DataAdapter** et **DataSet**
 - notification des erreurs pouvant survenir pendant la synchronisation des données
 - gestion des transactions sur une BD : **Transaction**

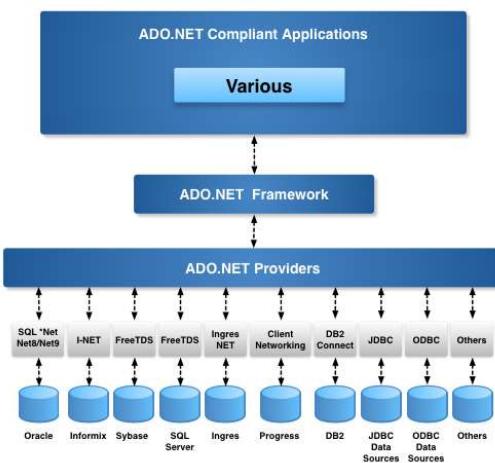


© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

8

Fournisseurs de données

- Depuis le Framework 3.5, il existe 4 types de fournisseurs :
 - ⇒ **SQL Server** : espace de nom `System.Data.SqlClient`, classe préfixée `Sql`
 - ⇒ **Oracle** : espace de nom `System.Data.OracleClient`, classe préfixée `Oracle`
 - ⇒ SQLServer et Oracle sont des fournisseurs de données entièrement managés
 - ⇒ **OLE DB** : espace de nom `System.Data.OleDb`, classes préfixé par `OleDb`, nécessite l'installation de MDAC (Microsoft Data Access Components)
 - ⇒ **OBDC** : espace de nom `System.Data.Odbc`, classe préfixée par `Odbc` nécessite MDAC, pilote ODBC au lieu d'une couche logicielle OLEDB



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit

Les espaces de noms

- **System.Data**
Classes de base de l'architecture ADO.NET permettant de construire des composants
- **System.Data.Common**
Classes partagées pour les fournisseurs de données managées de .NET (accès à une source de données, liaison aux objets DataSet)
- **System.Data.SqlTypes**
Classes permettant de représenter des types de données en mode natif SQL Server
- **System.Data.SqlClient**
Classes prenant en charge le fournisseur de données SQL Server de .NET (accès aux données pour SQL Server 7.0 et supérieur)
- **System.Data.OracleClient** : Oracle Managed Provider
Classes prenant en charge le fournisseur de données Oracle de .NET
- **System.Data.OleDb**
Classes pour le fournisseur OLE DB de .NET (accès managé pour tous les fournisseurs OLE DB pris en charge : SGBD Oracle, Jet, versions 6.5 et antérieur de SQL Server, ...)
- **System.Data.Odbc** : ODBC Managed Provider

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

Connexion / Déconnexion

- Les objets **xxxxConnection** permettent de créer une connexion à une source de données ou à un fournisseur de données (session unique) (OleDbConnection, OdbcConnection, SqlConnection, ...)

⇒ Utilisation de **chaînes de connexion** avec ConnectionString

```
SqlConnection maconnection = new SqlConnection();
maconnection.ConnectionString =
    "data source=Server3iL; database=db3iL; User Id='User'; pwd='Pass';Integrated Security=sspi";
```

⇒ Possibilité de préciser de nombreux paramètres :
Data Source (Serveur), Initial Catalog (Nom de la base), Integrated Security (identification), Persist Security Info (Login et pass visible), User ID (login), Pwd (mot de passe), temps d'attente (Connect Timeout), ...

Générateur sur : <http://www.connectionstrings.com>

⇒ Déconnexion par la méthode Close : `maconnection.Close();`

Chaîne de connexion à une base de données

```
SqlConnection maconnection = new SqlConnection();
maconnection.ConnectionString = "data source=172.16.2.15;" +
    "database=northwind; User Id='User'; pwd='Pass';Integrated Security=sspi";
```

```
OleDbConnection maoleDbConnection;
maoleDbConnection = new OleDbConnection();
maoleDbConnection.ConnectionString = @"Provider=Microsoft.Jet.OLEDB.4.0;" +
    "Password=1234;User ID=Admin;Data Source=D:\temp\test.mdb;Mode=Share Deny None;" +
    "Extended Properties='';Jet OLEDB:System database='';" +
    "Jet OLEDB:Registry Path='';Jet OLEDB:Database Password='';" +
    "Jet OLEDB:Engine Type=5;Jet OLEDB:Database Locking Mode=0;" +
    "...
```

```
string connectionString = "Driver={Microsoft Access Driver (*.mdb)};DBQ=C:\\Samples\\Northwind.mdb";
using (OdbcConnection connection = new OdbcConnection(connectionString))
{
    connection.Open();
}
```

Mode connecté / Mode déconnecté

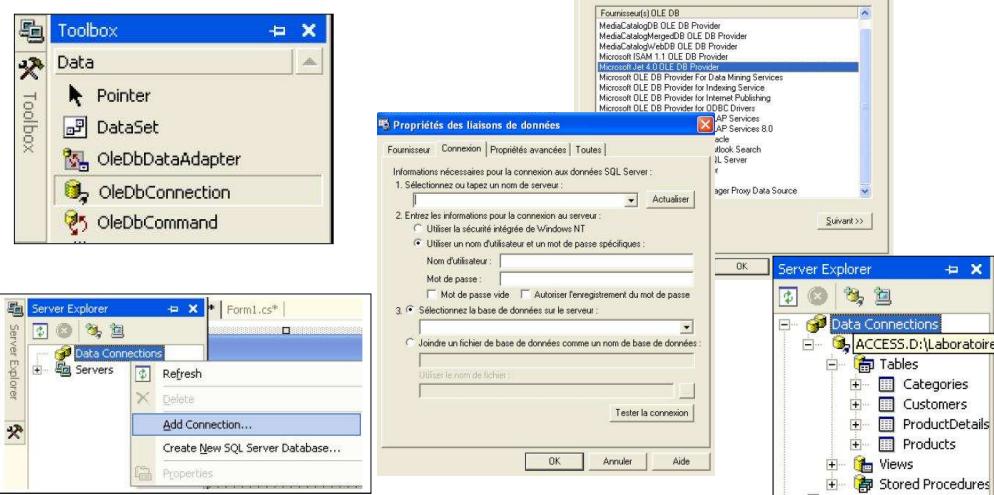
- ADO.NET permet de séparer les actions d'accès ou de modification d'une base de données.
- **Mode connecté : connexion permanente**
 - ⇒ données sont toujours à jour, gestion plus simple (connexion puis déconnexion)
 - ⇒ gaspille beaucoup de ressources entraînant des problèmes d'accès au réseau (tous les utilisateurs ont une connexion permanente avec le serveur même s'ils n'y font rien)
- **Mode déconnecté** : manipulation de bases de données **sans être connecté en permanence** (connexions pendant de courts laps de temps afin de faire les opérations) ⇒ utilisation de **DataSet**
 - ⇒ permet de connecter un nb important d'utilisateurs sur le même serveur
 - ⇒ meilleures performances des applications par la disponibilité des ressources
 - ⇒ données pas à jour en permanence ⇒ peut entraîner des conflits lors des mises à jour ⇒ code de gestion de conflits à prévoir

Mode connecté : Accès aux données

- **DataBinding** : consiste à prendre les données d'une source « provider » (DataGridView, ArrayList...) et de les placer, par un simple appel de méthode, dans un contrôle appelé « consumer » (DataGrid, DataList, DropDownList...).
⇒ la relation entre Provider et Consumer est appelée Binding
- **Command** : classe d'objet implementée chez chaque fournisseur qui permet d'exécuter des commandes SQL (Select, Update, ...) sur la base de données ouverte
- **DataReader** : object qui met directement à la disposition des utilisateurs un flux de données rapide, en lecture seule ⇒ rapidité de traitement
 - ⇒ accès par enregistrement ⇒ un seul enregistrement à la fois dans l'objet ⇒ pas d'opération complexe (tri ou accès direct)
 - ⇒ mode connecté ⇒ ferme l'objet après utilisation pour libérer la connexion
 - ⇒ déclinaison différente de l'objet pour chaque fournisseur : SqlDataReader, OracleDataReader, ...
 - ⇒ instantiation d'un DataReader par la méthode ExecuteReader de l'objet Command

Data Binding : connexion « graphique »

- Similaire aux Adodc de Ado



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

15

L'objet « Command »

- ⇒ permet d'exécuter les commandes sur la base de données connectée
- ⇒ chaque fournisseur de données de .NET possède sa propre version de l'objet **Command** (SqlCommand, OleDbCommand, ...)
- ⇒ présente d'autres méthodes : CreateCommand, CommandText (instruction SQL), CommandType (valant StoredProcedure, Text, ...)

```
OleDbConnection connection = new OleDbConnection("...");  
OleDbCommand command;  
command = new OleDbCommand("UPDATE MyTable SET MyField = 'MyVal'", connection);  
connection.Open();  
try  
{  
    rowAffected = command.ExecuteNonQuery();  
}  
finally  
{  
    connection.Close();  
}
```

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

16

Exemple d'un select avec un DataReader

- Exemple : Remplissage d'une « listbox » à partir d'une table :

```
SqlConnection connection = new SqlConnection("server=Srv-bd3il; database=northwind;
                                              User Id='User'; pwd='Pass';Integrated Security=sspi");
SqlCommand command = new SqlCommand("SELECT * FROM TABLE1",connection);
command.CommandType = CommandType.Text;
connection.Open();
SqlDataReader reader = command.ExecuteReader();
while (reader.Read())
{
    listBox1.Items.Add(reader.GetString(1));
}
reader.Close();
connection.Close();
```

Accès direct SQL Server : procédure stockée

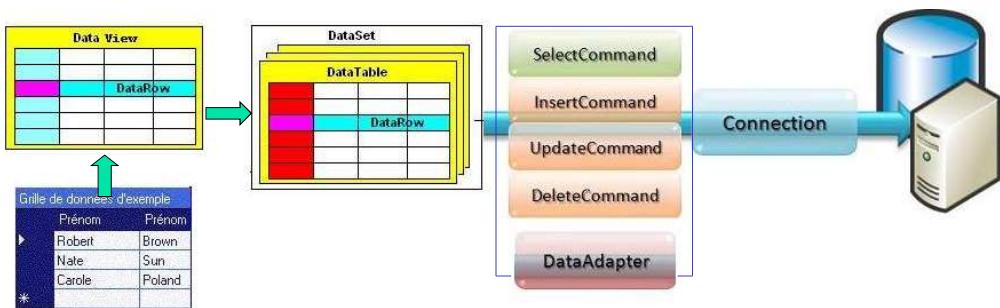
```
SqlCommand command = new SqlCommand("MaProcedureStockee",connection);
command.CommandType = CommandType.StoredProcedure;
SqlParameter myParam1 = new SqlParameter("@Param_1", SqlDbType.Int);
myParam1.Value = 100;
command.Parameters.Add(myParam1);
SqlParameter myParam2 = new SqlParameter("@Ending_Date", SqlDbType.DateTime);
myParam2.Value = "07/20/2005";
command.Parameters.Add(myParam2);
connection.Open();
SqlDataReader reader = command.ExecuteReader();
while (reader.Read())
{
    listBox1.Items.Add(reader.GetString(1));
}
reader.Close();
connection.Close();
```

Lorsqu'une procédure stockée renvoie des paramètres, les objets SqlParameter associés doivent avoir leur propriété Direction à Output

Mode déconnecté : Accès aux données

- **DataAdapter :**

- ⇒ objet contenant un ensemble de méthode permettant l'accès complet à la base de données comme SelectCommand, InsertCommand, UpdateCommand, DeleteCommand
- ⇒ déclinaison différente pour chaque fournisseur : **SqlDataAdapter**, ...
- ⇒ permet de travailler avec des « **Dataset** »



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

19

L'objet DataSet

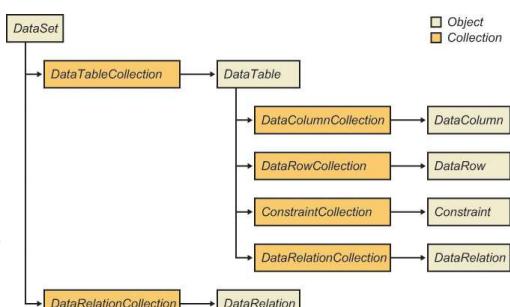
- **DataSet** : « **représentation des données en mémoire** » constituée d'une ou de plusieurs tables, elles-mêmes composées de colonnes et de lignes
 - ⇒ permet d'avoir « une base de données en mémoire » et d'y effectuer tout type d'opérations pour ensuite valider les opérations effectuées vers une BD

⇒ Apport majeur de ADO.Net qui supporte le **mode déconnecté**

- permet la sérialisation XML facilement (DataSet.WriteXml, ...)
- indépendant du modèle des données et indépendant de la source de données
- peut se créer, présente des objets tables, colonnes, lignes, champs, contraintes, ...

- Données synchronisées avec données BD via un objet **DataAdapter**

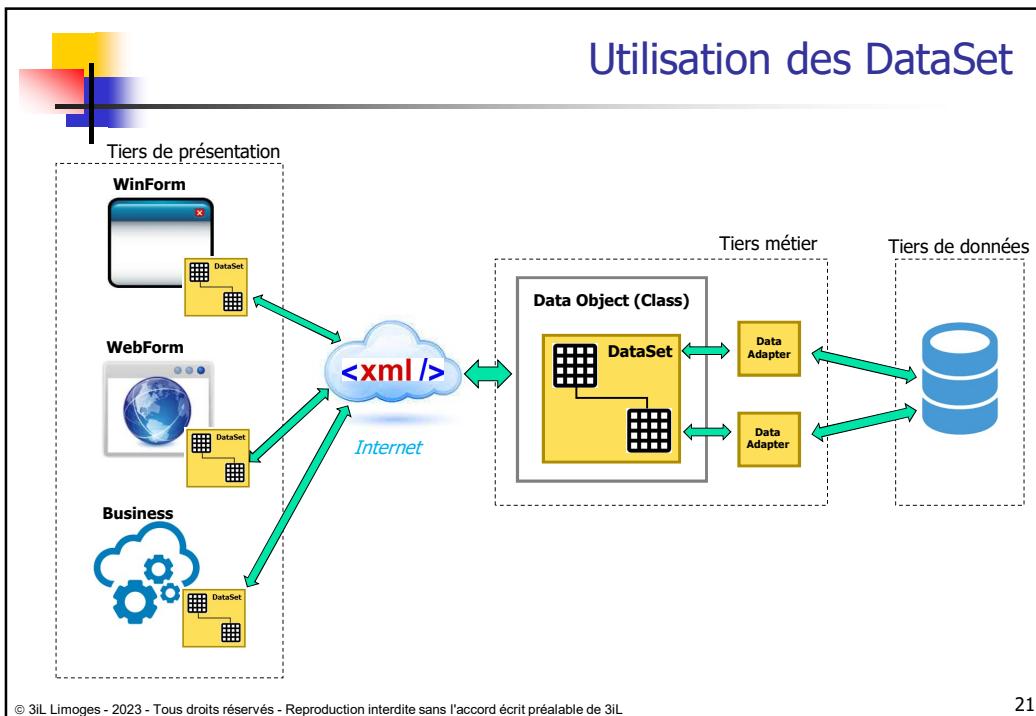
- Principe de remplir un **DataSet** par les données BD pour ensuite les manipuler
 - ⇒ créer une instance de classe **DataAdapter** qui réalisera le lien entre BD et **DataSet**
 - ⇒ remplir le **DataSet** par la méthode **Fill()**



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

20

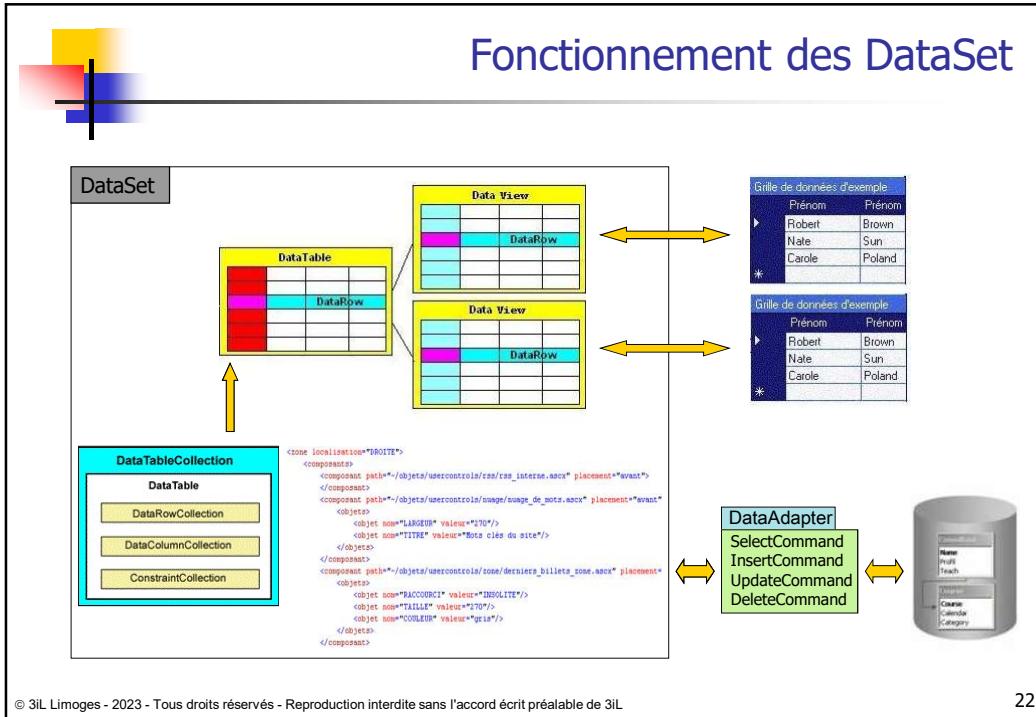
Utilisation des DataSet



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

21

Fonctionnement des DataSet



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

22

Remplir un DataGridView avec un DataSet

```
String MaBase = "";
SqlDataAdapter dAdapterMaBase;
SqlConnection connexion;
connexion = new SqlConnection("Data source = " + MaBase);
dAdapterMaBase =
    new SqlDataAdapter("SELECT * FROM TABLE1", connexion);
DataSet dSet = new DataSet();
try
{ connexion.Open(); }
catch
{
    MessageBox.Show("Impossible d'ouvrir la connexion", "Erreur");
    return;
}
dAdapterMaBase.Fill(dSet, "TAB1");
connexion.Close();
dataGridView1.DataSource = dSet.Tables["TAB1"];
dataGridView1.Update();
```

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

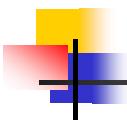
23

Exemple d'utilisation d'un DataSet

```
string requete, connexionString;
connexionString = @"Data Source=Server3il;Initial Catalog=DB3il;Integrated Security=true;";
requete = "SELECT * FROM Employe";
DataSet ds = new DataSet();
 IDbDataAdapter da = new SqlDataAdapter();
 IDbConnection connexion = new SqlConnection(connexionString);
 IDbCommand commande = connexion.CreateCommand();
 commande.CommandText = requete;
 commande.CommandType = CommandType.Text; // Choix de la commande executée par le DataAdapter
da.SelectCommand = commande;
da.Fill(ds); //Remplissage du DataSet
Console.WriteLine("Nom de la table : {0} | possède {1} enregistrement(s) \n\n\n",
    ds.Tables[0].Rows.Count);
//Affichage des informations du DataSet
foreach ( DataColumn colonne in ds.Tables[0].Columns) //Affichage des noms des colonnes
{   Console.Write(colonne.ColumnName + "\t");   }
Console.WriteLine("\n\n");
foreach ( DataRow ligne in ds.Tables[0].Rows) //Affichage des enregistrements
{   foreach ( DataColumn colonne in ds.Tables[0].Columns)
        {   Console.Write(ligne[colonne.ColumnName] + "\t");   }
    Console.WriteLine("\n");
}
Console.ReadLine();
```

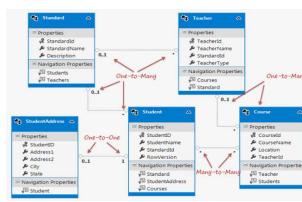
© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

24



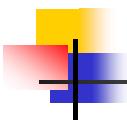

2. - Entity Framework, Linq



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

25

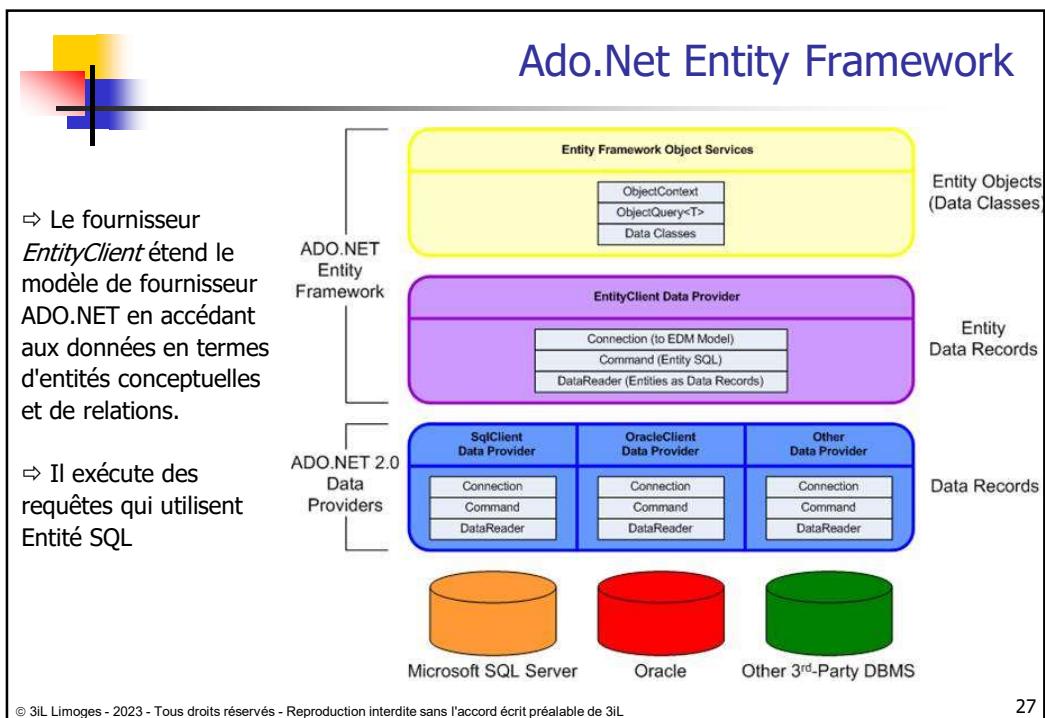


Ado.NET Entity Framework

- **ADO.NET Entity Framework :**
Framework de **mapping objet / relationnel** pour modéliser et manipuler les données non plus sous forme de tables, mais sous forme d'**entités**
- Modélisation selon 3 axes :
 - ⇒ **Conceptual model** (modèle conceptuel), appelé aussi EDM (Entity Data Model), qui décrit les entités (attributs et opérations) et les associations
 - ⇒ **Storage model** (modèle physique), qui décrit les tables (colonnes, références) et les procédures stockées
 - ⇒ **Mapping** entre le *conceptual model* et le *storage model*, qui décrit la correspondance (*mapping*) entre les tables et les entités
- Chacun de ces 3 points de vue est décrit sous forme d'un document XML
- ⇒ permet de créer des applications d'accès aux données en programmant par rapport à un **modèle d'application conceptuel** au lieu de programmer directement par rapport à un schéma de stockage relationnel
- ⇒ diminue la quantité de code et les besoins en maintenance pour les applications orientées objet
- ⇒ prise en charge de la fonctionnalité LINQ

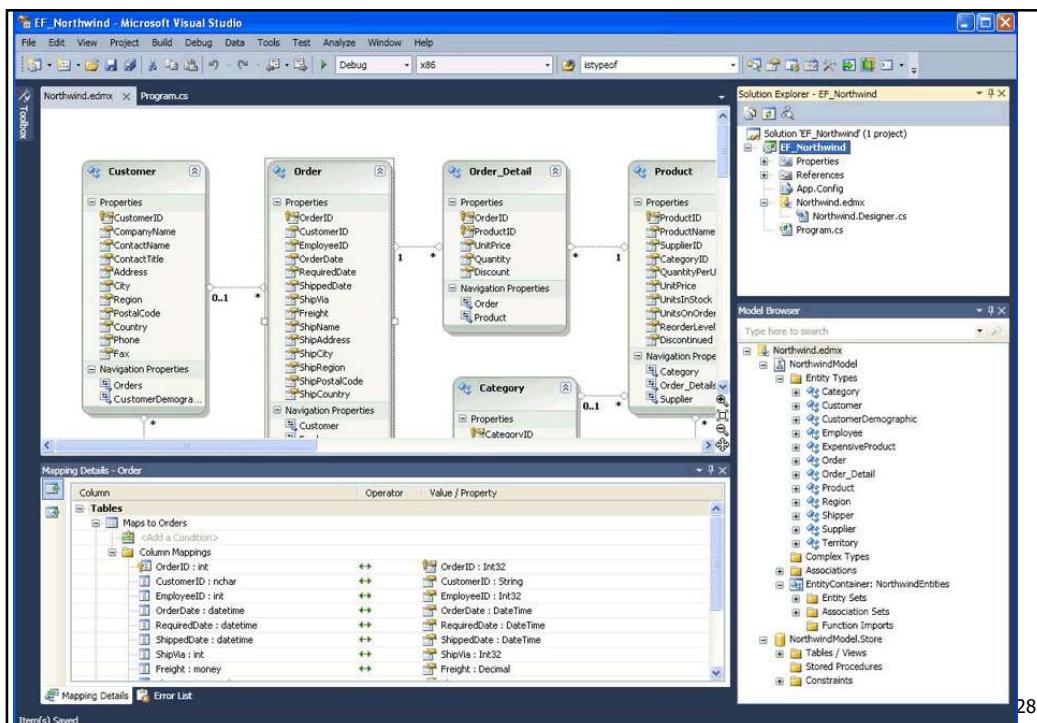
© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

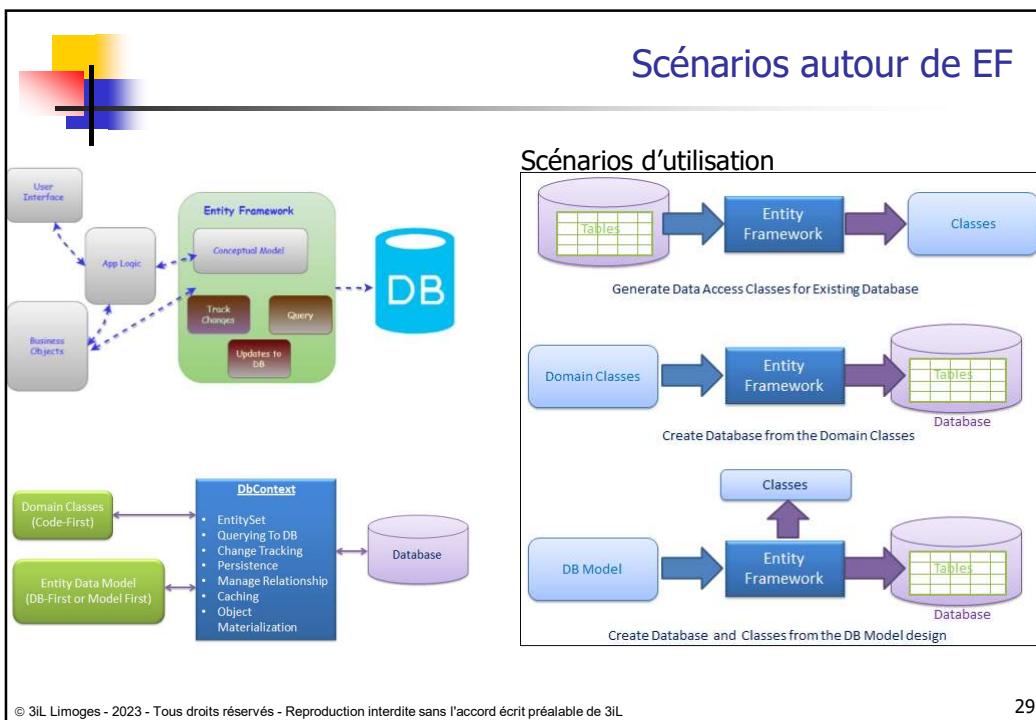
26



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

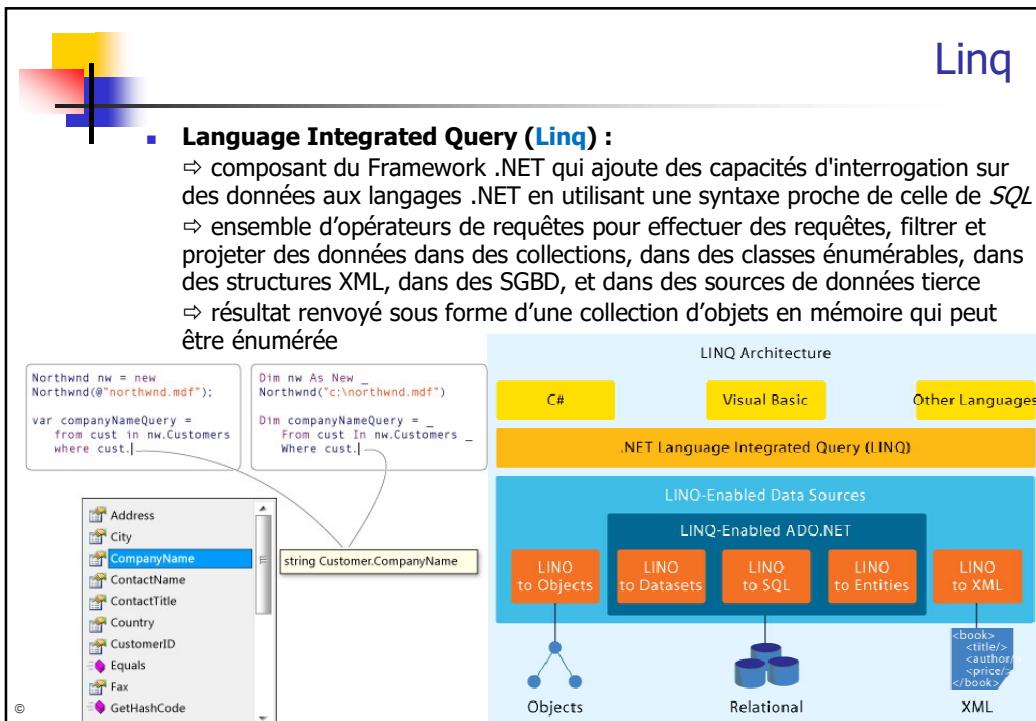
27

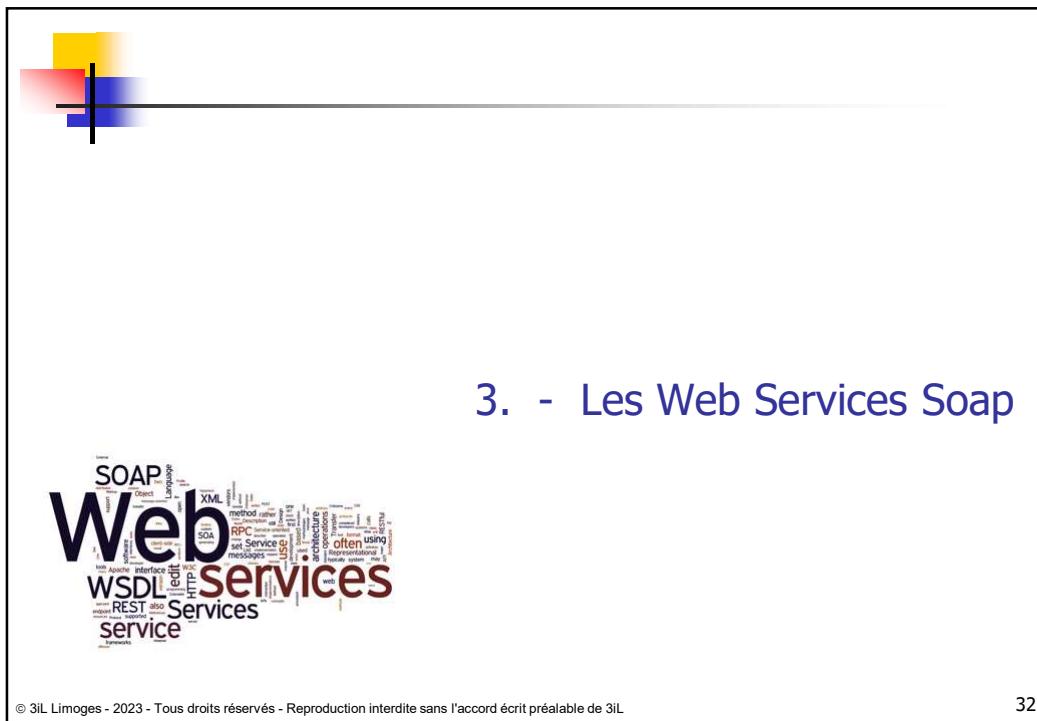
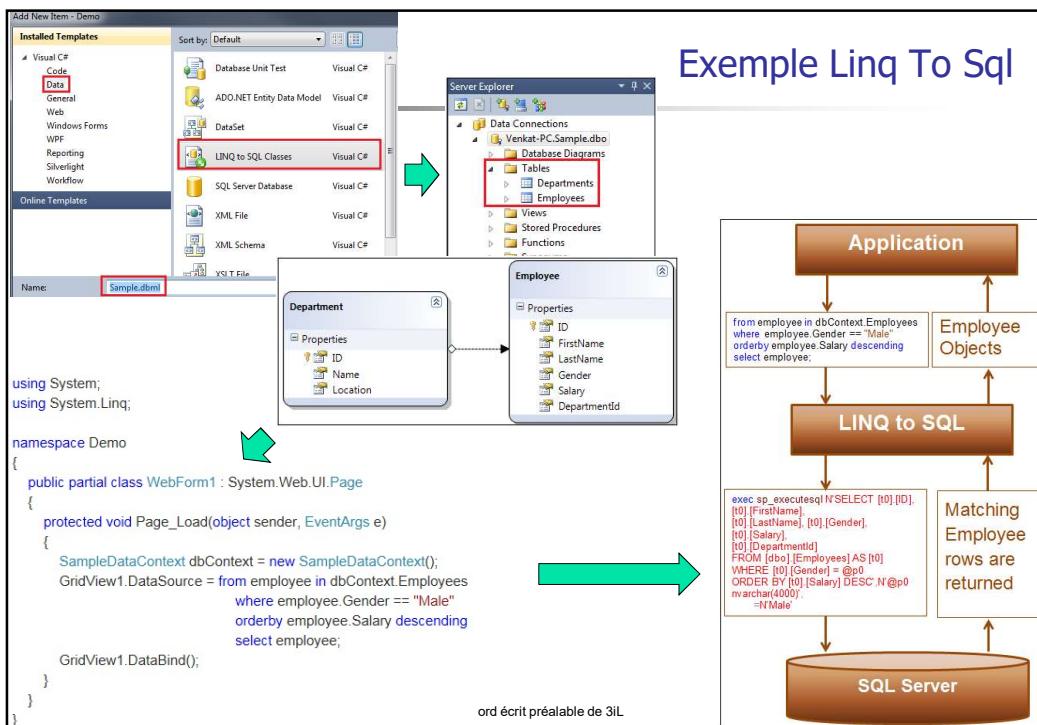




© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

29

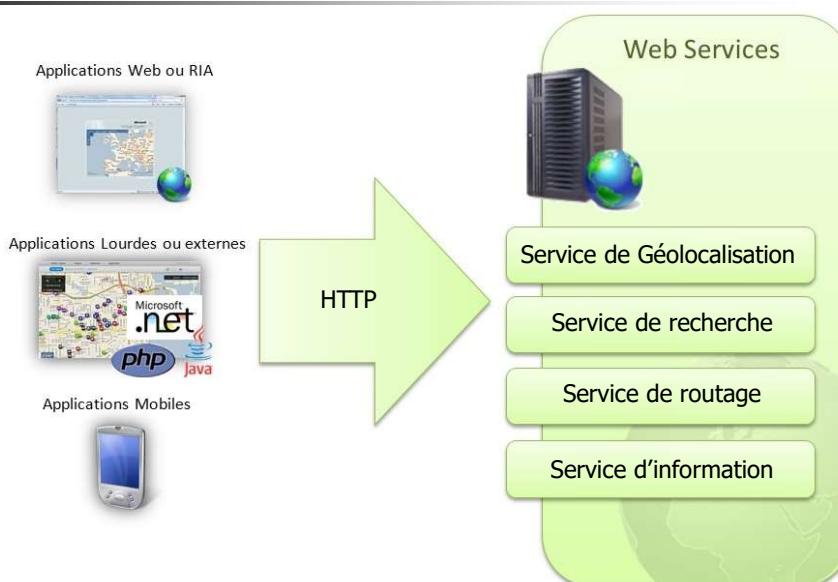




Services Web ou WebServices

- **Services Web** dits « **WebServices** » :
 - ⇒ Services applicatifs interrogeable via Internet sur le port http ou https (80/443)
 - ⇒ Les données sont transmises en XML (Soap) ou Json (Rest), voir en texte
- Un WebService exécute des **WebMéthodes** (= méthodes d'un WebService) demandées par le client puis renvoie les réponses sous forme de fichiers.
 - ⇒ Le client n'a plus qu'à récupérer cette réponse et en faire ce que lui semble bon.
- L'utilisation des WebServices avec les bases de données permet de :
 - ⇒ travailler en mode déconnecté : les Web Services ne renvoient que des données qui sont des « copies locales » d'une partie de la base de données
 - ⇒ accéder aux données à partir d'Internet (http/https)
 - ⇒ accéder à de multiples sources de donnée de différents SGBD
 - ⇒ de réutiliser des Web Service mis à disposition pour d'autres applications
- ⇒ En développement d'application, cela permet de mettre en place des **applications distribuées** autorisant un programme client d'effectuer des requêtes auprès du WebService qui renvoie au client une réponse.

Services Web

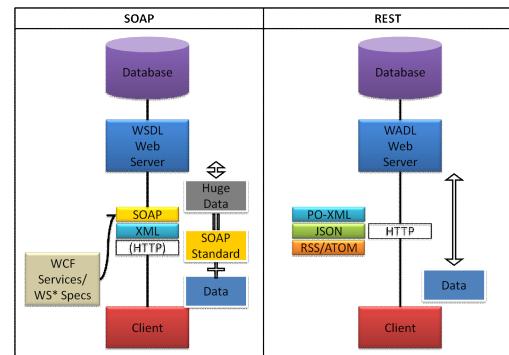


Services Web ou WebServices

- Le Framework .Net :**

- ⇒ Optimisé pour travailler avec des services SOAP, puis WCF (technologie MS)
- ⇒ Accepte les services Rest (Json)

	Web Service	WCF
Hébergement	Il peut être hébergé dans IIS	Il peut être hébergé dans IIS, auto-hébergement et Windows Service
Programmation	L'attribut [WebService] doit être ajouté à la classe.	L'attribut [ServiceContract] doit être ajouté à la classe.
Modèle	L'attribut [WebMethod] représente la méthode exposée au client.	L'attribut [OperationContract] représente la méthode exposée au client.
Opération	One-way, Request-Response sont les différentes opérations prises en charge dans le service Web	One-way, Request-Response. Duplex contient différents types d'opérations pris en charge dans WCF
XML	L'espace de nom System.Xml.Serialization est utilisé pour la sérialisation	L'espace de nom System.Runtime.Serialization est utilisé pour la sérialisation
Transports	Peut être accessible via HTTP, TCP, personnalisé	Peut être accessible via HTTP, TCP, canaux nommés, MSMQ, P2P, Custom
Codage	XML 1.0, MTOM (mécanisme d'optimisation du transmission de message), DIME, personnalisé	XML 1.0, MTOM, binaire, personnalisé

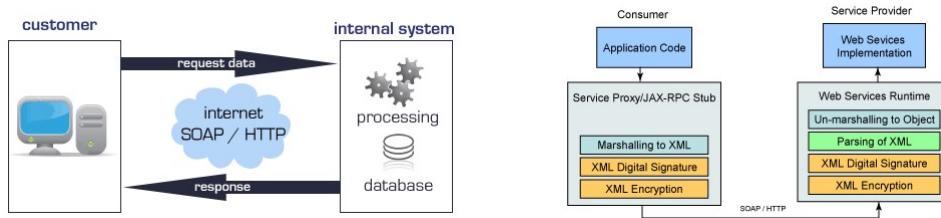


© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

35

WebServices Soap MS

- Les Web Services reposent sur les standards d'Internet : **HTTP**, **XML** et **SOAP**, et sur **IIS** (pour gérer les requêtes Http)
Protocoles standardisés ⇒ technologie exploitable par de nombreux langages
- La quasi totalité des langages informatiques supporte ces protocoles
⇒ Un Web Service provenant d'une plateforme .NET peut être utilisé via les langages Perl, PHP, Python, Delphi, Cobol ...
- Un WebService possède l'extension **.asmx** ou **.svc** (sous .NET avec IIS)
- Un WebService peut être sécurisé en ajoutant une authentification (classique, type Windows ou personnalisée) et/ou en intégrant SSL (HTTPS)



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

36

Le protocole SOAP

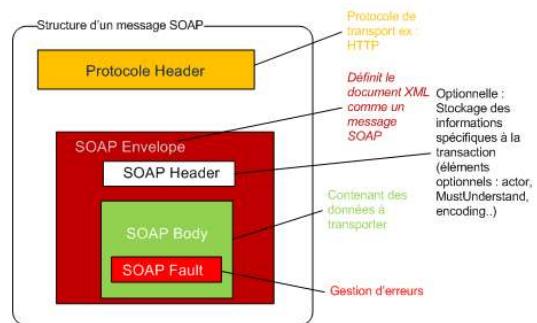
- **SOAP** : Simple Object Access Protocol (Protocole Simple d'Accès aux Objets)
consiste à faire circuler du XML via du http sur le port 80
XML est un langage standard et le port le plus utilisé est le port 80 (firewalls)

- Message SOAP

syntaxe décrite sur le site officiel W3C : <http://www.w3.org/TR/SOAP/>
un message SOAP est un document XML qui doit avoir la forme suivante :

- ⇒ 1 déclaration XML non obligatoire
⇒ 1 enveloppe SOAP composée de :
 1 en-tête SOAP (HEADER)
 + 1 corps SOAP (BODY)

Enveloppe :
déclarée via la balise <SOAP-ENV:Envelope>
et composée d'un corps <SOAP-ENV:Body>



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

37

Le protocole SOAP

- Ex. :

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">

    <SOAP-ENV:Body>
        <ns1:GetNombreResponse
            xmlns:ns1="urn:MySoapServices"
            SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <return xsi:type="xsd:int">10</return>
        </ns1:GetNombreResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">

    <SOAP-ENV:Body>
        <ns1:GetNombre
            xmlns:ns1="urn:MySoapServices">
            <param1 xsi:type="xsd:int">10</param1>
        </ns1:GetNombre>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

38

Contrat WSDL

- Chaque service web est identifié par un **contrat WSDL** qui représente la structure SOAP/XML du Service Web
Il faut faire référence au contrat WSDL du Service Web pour le consommer
- Le **WSDL** sert à décrire :
 - ⇒ le protocole de communication (SOAP RPC ou SOAP orienté message)
 - ⇒ le format de messages requis pour communiquer avec ce service
 - ⇒ les méthodes que le client peut invoquer
 - ⇒ la localisation du service

- Le contrat WSDL est un document XML qui est visible :

⇒ soit en cliquant sur le lien « Service description »
⇒ soit en ajoutant "**?WSDL**" à la fin de l'url du Service Web

<http://www.mon serviceweb.svc?WSDL>

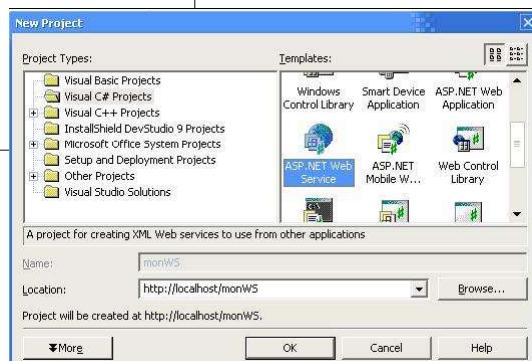
```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions xmlns:i="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://ws.cdyne.com/WeatherWS" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/http/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" targetNamespace="http://ws.cdyne.com/WeatherWS" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">*
  <wsdl:types>
    <xs:element formDefault="qualified" targetNamespace="http://ws.cdyne.com/WeatherWS">
      <xs:complexType />
      <s:element name="GetWeatherInformation">
        <s:complexType />
        <s:sequence>
          <s:element name="GetWeatherInformationResponse">
            <s:complexType />
            <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="GetWeatherInformationResult" type="tns:ArrayOfWeatherDescription" />
            </s:sequence>
          </s:element />
        </s:sequence>
      </s:element />
    </xs:element />
    <xs:complexType name="ArrayOfWeatherDescription">
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" name="WeatherDescription" type="tns:WeatherDescription" />
      </xs:sequence>
    </xs:complexType />
    <xs:complexType name="WeatherDescription">
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="1" name="WeatherID" type="s:short" />
        <xs:element minOccurs="0" maxOccurs="1" name="Description" type="s:string" />
        <xs:element minOccurs="0" maxOccurs="1" name="PictureURL" type="s:string" />
      </xs:sequence>
    </xs:complexType />
  </wsdl:types>
```

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite

Création d'un Web Service

```
using System.Web.Services;
...
[WebService(Namespace="http://mon_dommaine.com")]
public class MonService : System.Web.Services.WebService
{
    [WebMethod(Description="Description de la méthode")]
    public string MaMethode( )
    {
        return ValDeRetour;
    }
}
```

Se créer dans Visual Studio



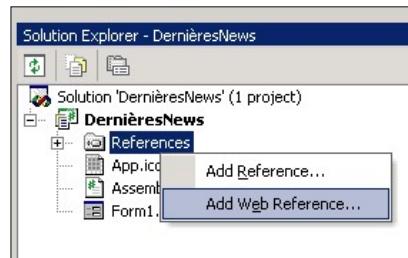
Pour vérifier si un WebService fonctionne, le tester sous Visual-Studio à l'aide d'Internet Explorer

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

Interrogation d'un Web Service

- **Pour utiliser un Web Service**
 - ⇒ ajouter une Web Référence pointant sur le service
 - ⇒ création d'une classe Proxy qui va faire la jonction entre le service Web et le code client et permettre d'accéder aux WebMéthodes du WebService
- Soit un Service Web local nommé **MonService.asmx**, qui présente une Web Method « **MaMéthode** » qui elle-même renvoie une valeur.
 - ⇒ Pour utiliser cette méthode, il faut donc ajouter une référence vers ce service et la nommer, par exemple « **MaRef** » dans notre cas, puis utiliser le code suivant :

```
MaRef.MonService srv = new MaRef.MonService();  
valRetour = srv.MaMéthode();
```



Sécurisation d'un service Web .Net

- **Authentification Windows** : sécurité intégrée dans Windows fournie par IIS et ASP.NET (authentification des applications Web)
 - 3 options d'authentification : Windows de base, Digest, intégrée Windows
 - Fichier Web.config : <authentication mode= "Windows"> </authentication>
 - ⇒ Windows de base : password et login codés mais non chiffrés
 - ⇒ intégrée Windows : utilise NTLM ou Kerberos (configuration du serveur)
 - ⇒ Windows Digest : utilise le hachage
- **Windows - de base sur SSL** : chiffrement SSL à configurer dans IIS, scénarios Internet, solution qui dégrade les performances
- **Windows - Certificats clients** : utilisation d'un certificat client, communication chiffrée par clés client et clés fournies par le serveur Web
- **Authentification d'en-têtes SOAP** : scénarios Internet sécurisés, informations d'identification de l'utilisateur passées dans l'en-tête SOAP
 - Le service Web doit spécifier qu'il attend l'en-tête SOAP contenant les informations d'identification et d'authentification et autoriser l'accès client
 - ⇒ Code spécifique : voir MSDN

Utiliser les DataSet dans les Services Web

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Web.Services;

[WebService(Namespace="http://microsoft.com/webservices/")]
public class Sample
{
    public SqlConnection connection = new SqlConnection("Data Source=(local);Integrated
                                                       Security=SSPI;Initial Catalog=Northwind");

    [WebMethod( Description = "Returns Northwind Customers", EnableSession = false )]
    public DataSet GetCustomers()
    {
        SqlDataAdapter adapter = new SqlDataAdapter(
            "SELECT CustomerID, CompanyName FROM Customers", connection);

        DataSet custDS = new DataSet();
        adapter.MissingSchemaAction = MissingSchemaAction.AddWithKey;
        adapter.Fill(custDS, "Customers");

        return custDS;
    }
}
```

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

43

Utiliser les DataSet dans les Services Web

```
[WebMethod( Description = "Updates Northwind Customers",
    EnableSession = false )]
public DataSet UpdateCustomers(DataSet custDS)
{
    SqlDataAdapter adapter = new SqlDataAdapter();

    adapter.InsertCommand = new SqlCommand(
        "INSERT INTO Customers (CustomerID, CompanyName) " +
        "Values(@CustomerID, @CompanyName)", connection);
    adapter.InsertCommand.Parameters.Add("@CustomerID", SqlDbType.NChar, 5, "CustomerID");
    adapter.InsertCommand.Parameters.Add("@CompanyName", SqlDbType.NChar, 15, "CompanyName");

    adapter.UpdateCommand = new SqlCommand(
        "UPDATE Customers Set CustomerID = @CustomerID, " +
        "CompanyName = @CompanyName WHERE CustomerID = " + "@OldCustomerID", connection);

    adapter.UpdateCommand.Parameters.Add("@CustomerID", SqlDbType.NChar, 5, "CustomerID");
    adapter.UpdateCommand.Parameters.Add("@CompanyName", SqlDbType.NChar, 15, "CompanyName");
    SqlParameter parameter = adapter.UpdateCommand.Parameters.Add(
        "@OldCustomerID", SqlDbType.NChar, 5, "CustomerID");
    parameter.SourceVersion = DataRowVersion.Original;
    adapter.Update(custDS, "Customers");

    return custDS;
}
```

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

44

Consommation du Service Web

```
using System;
using System.Data;

public class Client
{
    public static void Main()
    {
        Sample proxySample = new DsSample.Sample(); // Proxy object.
        DataSet customersDataSet = proxySample.GetCustomers();
        DataTable customersTable = customersDataSet.Tables["Customers"];

        DataRow row = customersTable.NewRow();
        row["CustomerID"] = "ABCDE";
        row["CompanyName"] = "New Company Name";
        customersTable.Rows.Add(row);

        DataSet updateDataSet = new DataSet();

        updateDataSet =
            proxySample.UpdateCustomers(customersDataSet.GetChanges());

        customersDataSet.Merge(updateDataSet);
        customersDataSet.AcceptChanges();
    }
}
```

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

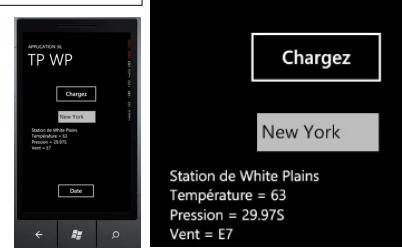
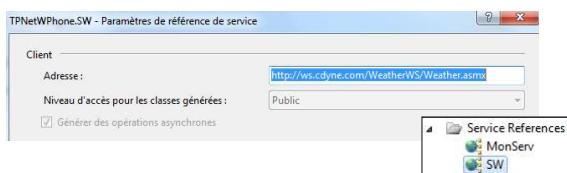
45

Appel d'un WS en mode désynchronisé

- Appel d'un WS en mode désynchronisé par un code Windows Phone

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    SW.WeatherSoapClient sr = new SW.WeatherSoapClient();
    sr.GetCityForecastByZIPCompleted += new EventHandler<SW.GetCityForecastByZIPCompletedEventArgs>(service_GetCityForecastByZIPCompleted);
    sr.GetCityWeatherByZIPCompleted += new EventHandler<SW.GetCityWeatherByZIPCompletedEventArgs>(service_GetCityWeatherByZIPCompleted);
    sr.GetCityForecastByZIPAsync("10278"); // New York
    sr.GetCityWeatherByZIPAsync("10278");
}

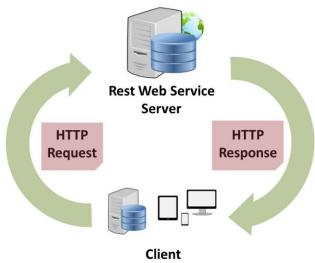
public void service_GetCityWeatherByZIPCompleted(object sender, SW.GetCityWeatherByZIPCompletedEventArgs e)
{
    tBk1.Text = "Station de " + e.Result.WeatherStationCity.ToString() + Environment.NewLine;
    tBk1.Text = tBk1.Text + "Température = " + e.Result.Temperature.ToString() + Environment.NewLine;
    tBk1.Text = tBk1.Text + "Pression = " + e.Result.Pressure.ToString() + Environment.NewLine;
    tBk1.Text = tBk1.Text + "Vent = " + e.Result.Wind.ToString() + Environment.NewLine;
}
```



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

46

4. - Les Web Services Rest



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

47

Différences entre REST et SOAP

- **REST** (Representational State Transfer) : style d'architecture qui repose sur le protocole **HTTP** :
 - ⇒ On accède à une ressource (par son URI unique) pour procéder à diverses opérations (GET lecture / POST écriture / PUT modification / DELETE suppression)
 - ⇒ opérations supportées nativement par HTTP
 - ⇒ Exemple : point de terminaison : <http://service/livre/>
 - **GET** : <http://service/livre/idLivre> (idLivre : identifiant du livre à lire)
 - **POST** : <http://service/livre/> (corps message : contenu du livre à créer)
 - **PUT** : <http://service/livre/idLivre> (corps message : contenu du livre à modifier)
 - **DELETE** <http://service/livre/idLivre> (corps message : contenu du livre à supprimer)
 - ⇒ REST est lisible (pas d'enveloppe XML ou autre) et facile à tester (un navigateur suffit), facile à coder et à déployer (script PHP, Java, C#, ...)
 - ⇒ REST n'impose ni ne revendique un format d'échange entre client et serveur
 - ⇒ représentation libres des données : XML, JSON, PHP sérialisé, MessagePack, ...
- Ex d'Url : http://monserveur/rest/?method=nom_methode¶metre=param_methode
<http://monserveur/rest/?method=modifervaleur&id=123>

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

48

Exemple de service Rest en C#

```
IRestService.svc.cs
using System.ServiceModel;
using System.ServiceModel.Web;
using System;

namespace RestService
{
    [ServiceContract]
    public interface IRestServiceMeteo
    {
        [OperationContract]
        [WebInvoke(Method = "GET",
            ResponseFormat = WebMessageFormat.Json,
            BodyStyle = WebMessageBodyStyle.Wrapped,
            UriTemplate = "json/{id}")]
        string JSONData(string id);

        [OperationContract]
        [WebInvoke(Method = "GET",
            ResponseFormat = WebMessageFormat.Xml,
            BodyStyle = WebMessageBodyStyle.Wrapped,
            UriTemplate = "xml/{id}")]
        string XMLData(string id);
    }
}
```

```
RestService.cs
using System;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;
using System.Threading.Tasks;

namespace RestService
{
    public class RestServiceMeteo : IRestServiceMeteo
    {
        public string JSONData(string id)
        {
            return "La valeur reçue est " + id ;
        }

        public string XMLData(string id)
        {
            return "La valeur reçue est " + id;
        }
    }
}

<behaviors>
<serviceBehaviors>
<behavior name="ServiceBehaviour">
    <!-- To avoid disclosing metadata information, set the
        serviceMetadata httpGetEnabled="true"/>
    <!-- To receive exception details in faults for debugging purposes, set
        serviceDebug includeExceptionDetailInFaults="false"/>
</behavior>
</serviceBehaviors>
<endpointBehaviors>
<behavior name="web">
    <webHttp/>
</behavior>
</endpointBehaviors>
</behaviors>
```

Web.config

```
<system.serviceModel>
<services>
<service name="RestService.RestService" behaviorConfiguration="ServiceBehaviour">
    <endpoint address="" binding="webHttpBinding" contract="RestService.IRestService" behaviorConfiguration="web">
    </endpoint>
</service>
</services>
```

© 3iL Limoges

49

Exemple de service Rest en C#

- Installation sur serveur IIS :

- <http://localhost:35798/ServiceRest.svc/json/23>

- <http://localhost:35798/ServiceRest.svc/xml/23>

```
<?xml version="1.0"?>
<XMLDataResponse xmlns="http://tempuri.org/">
    <XMLDataResult>La valeur reçue est 23</XMLDataResult>
</XMLDataResponse>
```

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

50

JSON

- **JSON** (JavaScript Object Notation) : (notation objet issue de JavaScript)
 - ⇒ format léger d'échange de données, facile à lire ou à écrire
 - ⇒ basé sur un sous-ensemble du langage JavaScript (un format texte complètement indépendant de tout langage (considéré comme un langage d'échange de données idéal))
- ⇒ Un **document JSON** ne comprend que 2 types d'éléments structuels :
 - des ensembles de paires « nom » (alias «clé») / « valeur »
 - des listes ordonnées de valeurs
- ⇒ Les **éléments** qui représentent 3 types de données :
 - des objets
 - des tableaux
 - des valeurs génériques de type tableau, objet, booléen, nombre, chaîne ou null

```
{ "Title": "The Cuckoo's Calling", "Author": "Robert Galbraith", "Genre": "classic crime novel", "Detail": { "Publisher": "Little Brown", "Publication_Year": 2013, "ISBN-13": "9781408704004", "Language": "English", "Pages": 494 } } [ { "type": "Hardcover", "price": 16.65 }, { "type": "Kindle Edition", "price": 7.03 } ] }
```

The diagram illustrates a JSON object structure with various annotations:

- Object Starts: Indicated by blue arrows pointing to the opening brace '{' at the beginning of the object and after the 'Detail' key.
- Value string: Indicated by a green arrow pointing to the string value 'Little Brown'.
- Value number: Indicated by a green arrow pointing to the number value '2013'.
- Object ends: Indicated by blue arrows pointing to the closing brace '}' after the 'Detail' object and after the second object in the array.
- Array starts: Indicated by green arrows pointing to the opening brace '[' before the array and after the first object in the array.
- Object Starts: Indicated by blue arrows pointing to the opening brace '{' before the first object in the array and after the second object in the array.
- Object ends: Indicated by blue arrows pointing to the closing brace '}' after the first object in the array and after the second object in the array.
- Array ends: Indicated by green arrows pointing to the closing brace ']' after the array.

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

51

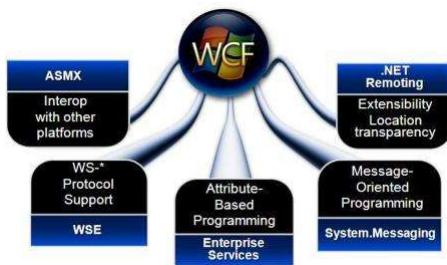
Différences entre REST et SOAP

- Quelles sont les différences entre SOAP et REST ?
 - ⇒ REST et SOAP sont deux architectures utilisées pour fournir des services web
 - ⇒ style d'architecture basé sur le protocole HTTP
- **REST** (Representational State Transfer) : ne repose que sur le protocole **HTTP**
 - ⇒ souvent utilisé lorsque la simplicité de mise en œuvre est recherchée
 - ⇒ REST est lisible (pas d'enveloppe XML superflue)
 - ⇒ REST est facile à tester (un navigateur suffit) et facile de mise en œuvre (un script PHP classique peut souvent être considéré comme RESTful)
 - ⇒ REST n'est pas sécurisé : attention aux paramètres critiques dans les Url
- **SOAP** (Simple Object Access Protocol) : protocole d'échange bâti sur **HTTP/XML**
 - ⇒ très utilisé pour les WebServices et intégré dans de nombreux outils de dév.
 - ⇒ impose une enveloppe contenant des informations et un modèle de données
 - ⇒ possibilité d'export de classes en Webservices, de génération automatique de clients à partir des WSDL, de contrôles forts sur les types de données attendus
 - ⇒ Soap n'est pris en compte dans divers Framework de développement client (Framework JS/HTML mobile, Qt, ...)

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

52

5. - Les services WCF



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

53

Windows Communication Foundation (WCF)

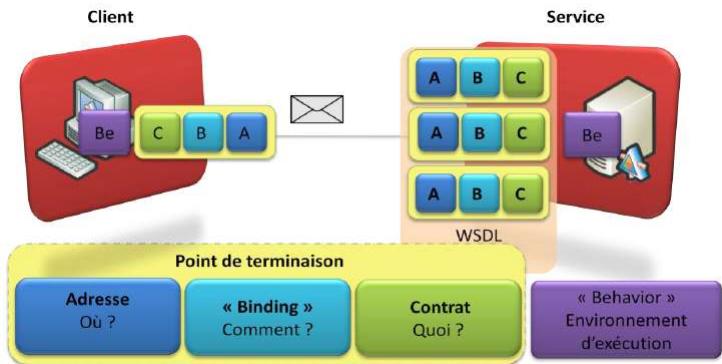
- **WCF** : nouvelle couche de communication introduit par le Framework 3.0
 - ⇒ couche d'abstraction pour la construction d'applications orientées services qui unifie et simplifie la mécanique d'intégration des services communiquant (services Web, Remoting, COM+, Microsoft Transaction Server, Microsoft Message Queuing)
 - ⇒ couche créée afin d'unifier les différents modèles d'écritures d'applications « communicantes »
 - ⇒ permet la communication entre applications sans pour autant devoir adapter les développements (HTTP/SOAP ou TCP/binaire sans adapter de code)
- Pour exécuter et rendre accessible aux clients un service WCF :
 - ⇒ l'héberger dans une application Hôte, un service Windows ou par IIS
 - ⇒ Extension **.SVC**
- WCF utilise des messages SOAP pour les communications entre processus (XML si dialogue process WCF avec non WCF, format binaire si WCF-WCF)
- référence au namespace **System.ServiceModel**

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

54

Structure de WCF

- Un service WCF peut exposer un ou plusieurs **points de communication** (« endpoint »), étant constitué d'une adresse (A), d'un binding (B) et d'un contrat (C):
 - ⇒ L' « **adresse** » spécifie la localisation du service, définie par une URI renseignée dans le code ou déclarée dans le fichier de configuration
 - ⇒ Le « **binding** » permet de décrire le protocole utilisé pour se connecter à un point de communication
 - ⇒ Le « **contrat** » permet au service de décrire explicitement la liste des opérations qu'il expose
 - ⇒ Le « **behavior** » permet de paramétriser l'environnement d'exécution du service



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

55

Contrats WCF

- Un service WCF s'appuie sur 2 types de contrats, 1 contrat de données et 1 contrat de service
 - ⇒ **contrat de données** : définit le format des données échangées d'un objet métier défini par l'attribut [DataContract] et ses membres par [DataMember]
 - ⇒ **contrat de service** : définit les méthodes (= opérations) exposées aux clients par le service WCF : attribut [ServiceContract] et ses membres par [OperationContract]

```
namespace WcfService1
{
    [ServiceContract]
    public interface IService1
    {
        [OperationContract]
        string GetData(int value);

        [OperationContract]
        CompositeType GetDataUsingDataContract(CompositeType composite);
    }
}
```

```
[DataContract]
public class CompositeType
{
    bool boolValue = true;
    string stringValue = "Hello ";

    [DataMember]
    public bool BoolValue
    {
        get { return boolValue; }
        set { boolValue = value; }
    }

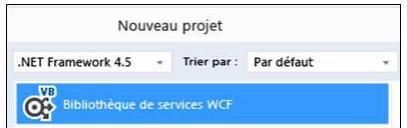
    [DataMember]
    public string StringValue
    {
        get { return stringValue; }
        set { stringValue = value; }
    }
}
```

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

56

Création d'un Service WCF en C#

Création d'un projet WCF en C#



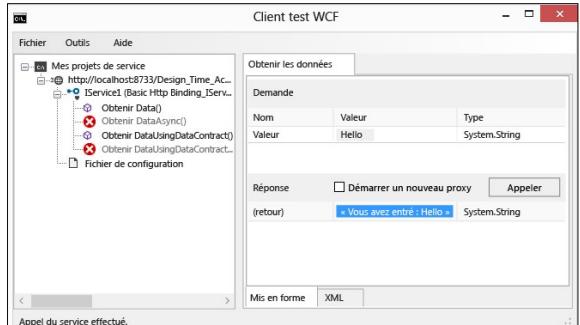
IService.svc.cs

```
[OperationContract]
string GetData(string value);
```

Service.cs

```
public string GetData(string value)
{
    return string.Format("You entered: {0}", value);
}
```

Test du service dans l'outil de VS

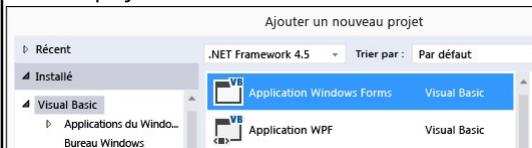


© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

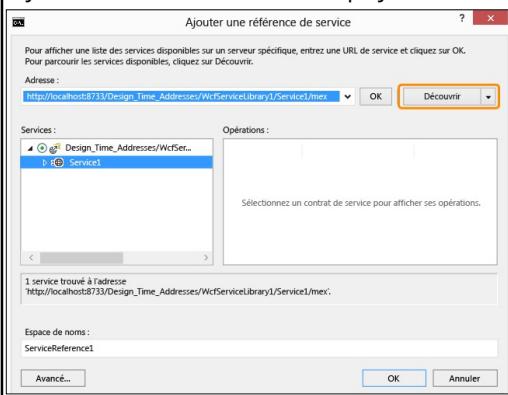
57

Utilisation du service dans un client

Création projet



Ajout de la référence de service au projet

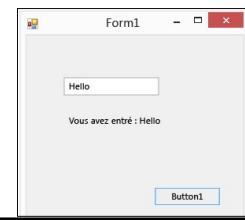


Appel en VB

```
Private Sub Button1_Click(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles Button1.Click  
    Dim client As New ServiceReference1.Service1Client  
    Dim returnString As String  
  
    returnString = client.GetData(textBox1.Text)  
    Label1.Text = returnString  
End Sub
```

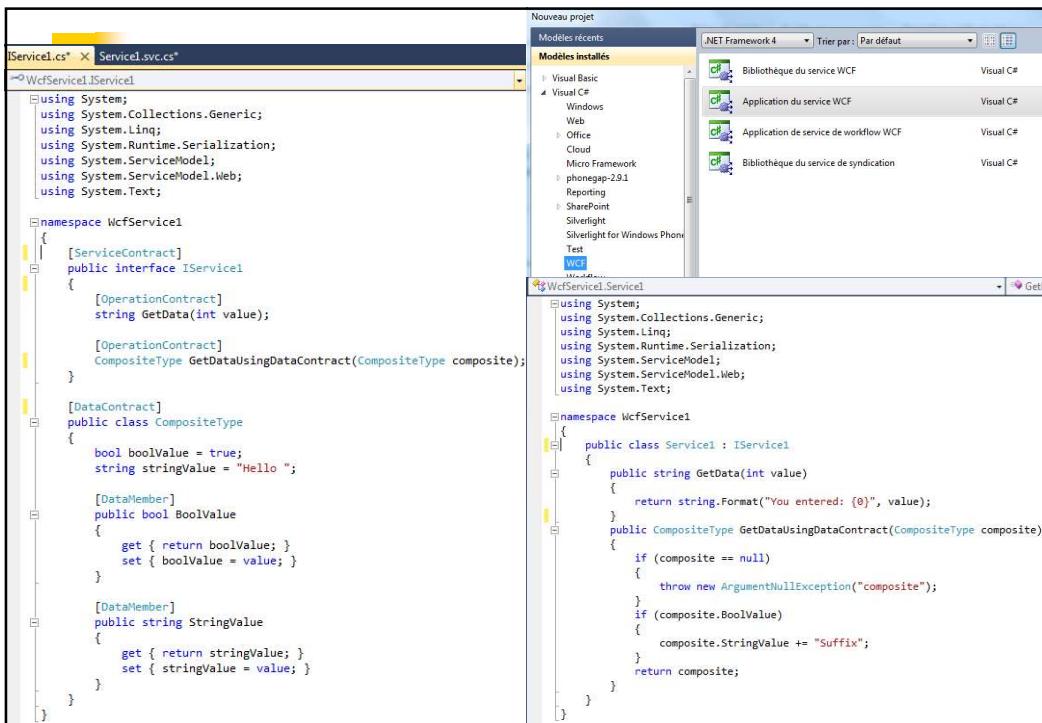
Appel en C#

```
private void button1_Click(System.Object sender, System.EventArgs e)  
{  
    ServiceReference1.Service1Client client = new  
        ServiceReference1.Service1Client();  
    string returnString;  
  
    returnString = client.GetData(textBox1.Text);  
    label1.Text = returnString;  
}
```



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

58



Que faut-il retenir ?

- Définir ADO.Net, les fournisseurs d'accès managé et non managé, les classes préfixée identifiant le fournisseur
- Différencier mode connecté et déconnecté
- Connaître les 3 méthodes d'accès du mode connecté
- Définir un DataSet, sa structure, le rôle d'un DataAdapter
- Expliquer le mapping objet / relationnel avec entity Framework :
- Définir ce qu'est un WebService (WebMéthode, protocole)
- Définir ce qu'est un WebService SOAP (protocole SOAP, WSDL, XML)
- Définir ce qu'est un WebService REST (url, simplicité, Json)
- Expliquer la technologie de WebService WCF

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

60

Annexes

Liens pour aller plus loin

<http://msdn.microsoft.com>
<http://dotnet.developpez.com/>
<http://dotnet-france.com/>
<http://www.dotnetguru.org/>

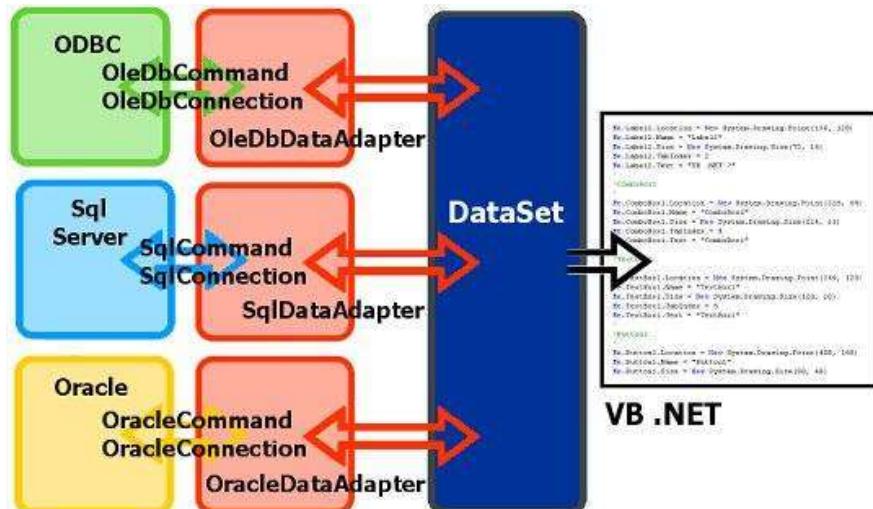
ADO.NET :

[http://msdn.microsoft.com/fr-fr/library/e80y5yhx\(VS.80\).aspx](http://msdn.microsoft.com/fr-fr/library/e80y5yhx(VS.80).aspx)
Utilisation de DataSets dans ADO.NET
[http://msdn.microsoft.com/fr-fr/library/ss7fbaez\(VS.80\).aspx](http://msdn.microsoft.com/fr-fr/library/ss7fbaez(VS.80).aspx)

Service Web :

<http://msdn.microsoft.com/fr-fr/library/7bkzywba.aspx>

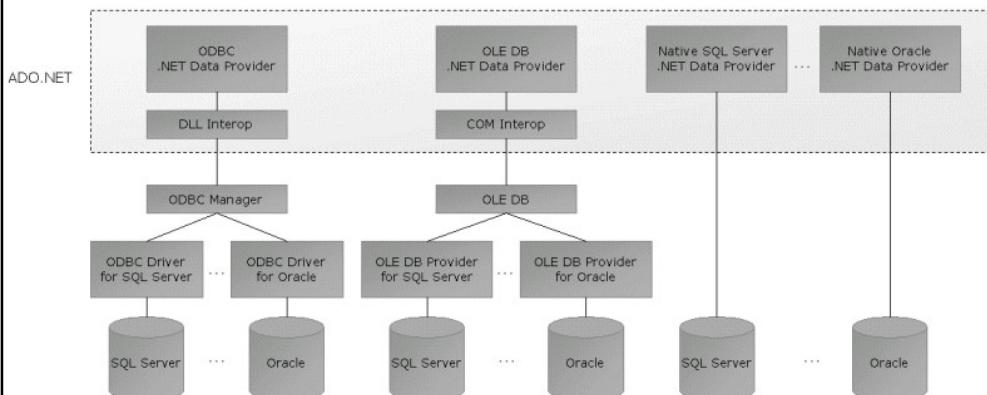
Classes de bases d'accès aux données



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

63

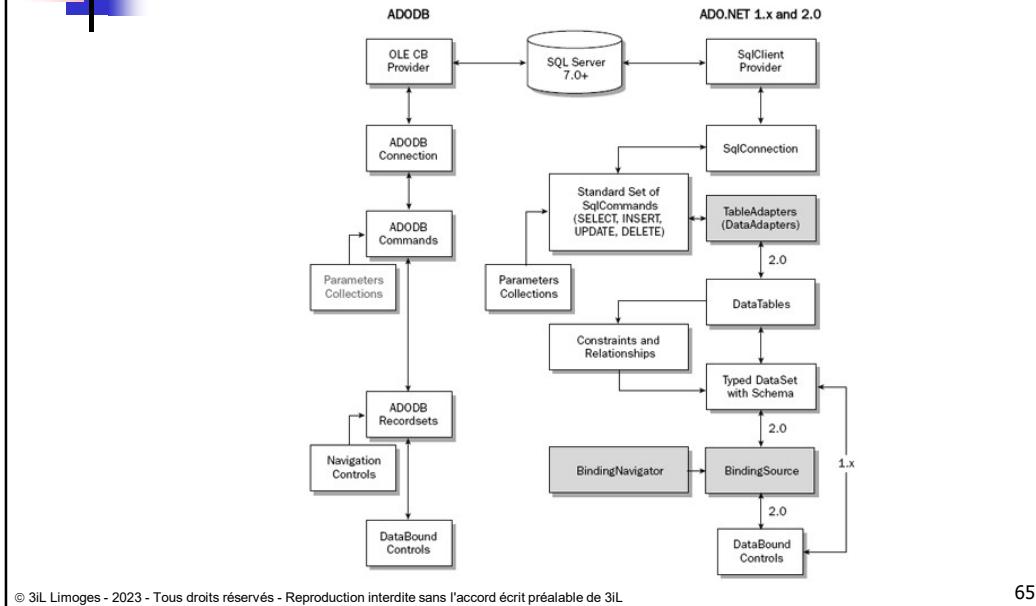
ADO.Net et les fournisseurs d'accès



© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

64

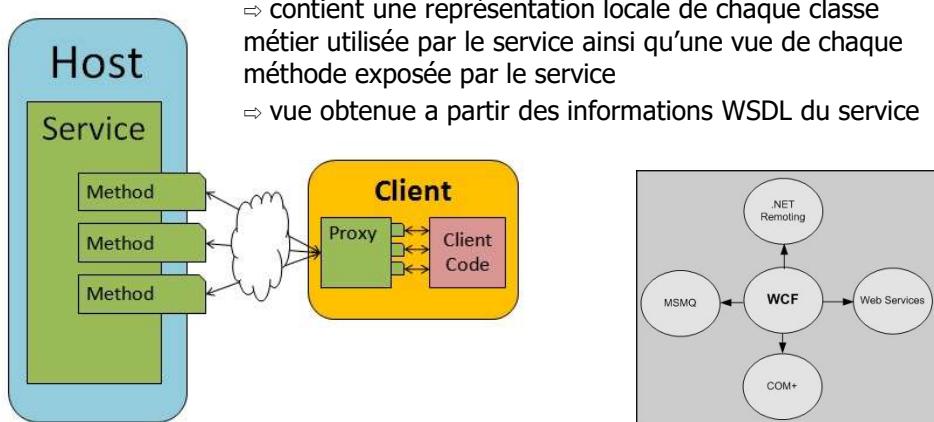
ADODB ≠ ADO.Net



65

Classe Proxy

- **Classe proxy :** classe utilisée côté client pour masquer la complexité des communications entre le service WCF et l'application qui l'utilise.
 - ⇒ contient une représentation locale de chaque classe métier utilisée par le service ainsi qu'une vue de chaque méthode exposée par le service
 - ⇒ vue obtenue à partir des informations WSDL du service



66

Connaitre les providers disponibles

- Le méthode **GetFactoryClasses** de la classe *DbProviderFactories* (*namespace System.Data.Common*) permet de donner les fournisseurs disponibles sur un poste de travail

```
Sub Main()
    Dim listeFournisseur As DataTable
    listeFournisseur = DbProviderFactories.GetFactoryClasses()

    For Each colonne As DataColumn In listeFournisseur.Columns
        Console.WriteLine(colonne.ColumnName + vbTab)
        ' Affiche le nom des colonnes
    Next
    Console.WriteLine(vbNewLine + vbNewLine)

    For Each ligne As DataRow In listeFournisseur.Rows
        ' Affiche chaque ligne
        For Each colonne As DataColumn In listeFournisseur.Columns
            ' Affiche les cellules
            Console.WriteLine(ligne.Item(colonne.ColumnName) + vbTab)
        Next
        Console.WriteLine(vbNewLine + vbNewLine) ' Retour à la ligne
    Next
    Console.WriteLine(vbNewLine + vbNewLine)
End Sub
```

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

67

Authentification de base

```
public class MyIdentity
{
    public string AuthenticationType;
    public bool IsAuthenticated;
    public string Name;
}

[WebMethod]
public MyIdentity GetAuth()
{
    MyIdentity id=new MyIdentity();
    id.AuthenticationType=User.Identity.AuthenticationType;
    id.IsAuthenticated=User.Identity.IsAuthenticated;
    id.Name=User.Identity.Name;
    return id;
}

private void button1_Click(object sender, System.EventArgs e)
{
    localhost.Service1 sw=new localhost.Service1();
    CredentialCache cc=new CredentialCache();
    cc.Add(new Uri(sw.Url), "Basic", new NetworkCredential("ms","ms"));
    sw.Credentials=cc;
    localhost.MyIdentity mi=sw.GetAuth();
    label1.Text=mi.Name;
    label2.Text=mi.AuthenticationType;
    label3.Text=mi.IsAuthenticated.ToString();
}
```

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

68