



COURS .NET DATASET - LINQ

Lemettre Arnaud
Arnaud.lemettre@gmail.com

SOMMAIRE



- ➡ Introduction
- ➡ Dataset (ado.net, data Set Typé)
- ➡ Linq
- ➡ Asp.net (composants, asmx)

INTRODUCTION

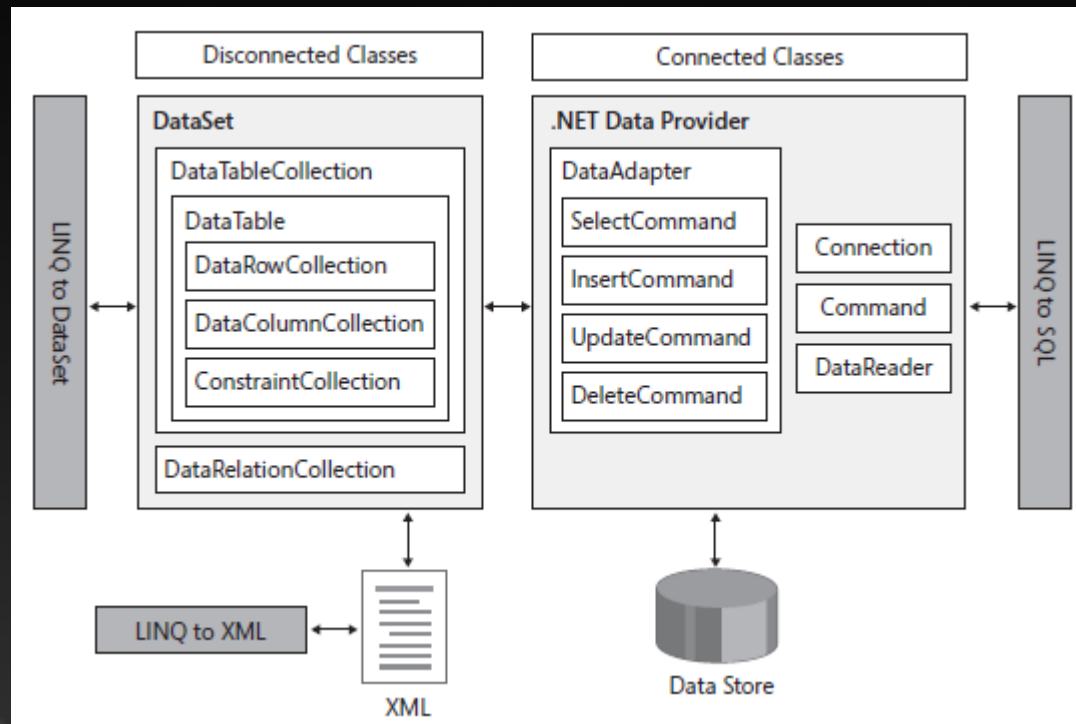


- ➡ La part la plus importante des applications métiers est certainement la gestion des données. C'est une part importante de la composition du framework .Net depuis sa création. Pour cela, on se sert essentiellement d'ado.net
- ➡ Il y a plusieurs modes de connexion :
 - ▶ Connecté
 - ▶ Déconnecté

INTRODUCTION



Description des classes utiles du framework, pour la gestion des données.





UTILISATION DES DATASETS

UTILISATION DES DATASETS



Connexion à la base :

```
SqlConnection connection = new SqlConnection();  
connection.ConnectionString = @"Data Source=localhost\sqlserver2008;Initial  
Catalog=TP;Integrated Security=True";
```

En ayant les bons providers on peut se connecter à n'importe quel type de base de données.

Pour avoir la bonne chaîne de connexion, vous pouvez utiliser le site suivant :

<http://connectionstrings.com/>

Avoir une chaîne de connexion dans le code... c'est bien, mais peu réutilisable, en effet il vaut mieux utiliser le web.config. Pourquoi ?

Lors d'un changement d'environnement, passage du développement à la production les serveurs de base de données ne sont pas les mêmes.

L'utilisation du web.config permet de changer de chaîne de connexion sans avoir à recompiler.

UTILISATION DES DATASETS



Utilisation du web.config, pour ajouter la connection String :

```
<?xml version="1.0"?>
<configuration>
  <appSettings/>
  <connectionStrings>
    <add name="TPConnectionString" connectionString="Data
Source=localhost\\sqlserver2008;Initial
Catalog=TP;Integrated Security=True"
        providerName="System.Data.SqlClient" />
  </connectionStrings>
  <system.web/>
  <system.webServer/>
</configuration>
```

UTILISATION DES DATASETS



Utilisation au sein du code :

```
using (SqlConnection connection = new
    SqlConnection(ConfigurationManager.ConnectionStrings
        ["TPConnectionString"].ConnectionString))
{
}
}
```

L'utilisation du using permettra de libérer la ressource une fois la connexion finie.

UTILISATION DES DATASETS



Utilisation au sein du code :

```
Public Static List<String> GetData()
{
    List<string> res = new List<string>();
    using (SqlConnection connection = new
        SqlConnection(ConfigurationManager.ConnectionStrings["TPConnectionString"].ConnectionString))
    {
        SqlDataAdapter adapter = new SqlDataAdapter();
        adapter.TableMappings.Add("Table", "T_Table1");
        connection.Open();

        SqlCommand command = new SqlCommand(
            "SELECT * FROM dbo.T_Table1;",
            connection);
        command.CommandType = CommandType.Text;

        adapter.SelectCommand = command;

        DataSet dataSet = new DataSet("T_Table1");
        adapter.Fill(dataSet);
        DataTable table = dataSet.Tables["T_Table1"];
        foreach (DataRow item in table.Rows)
        {
            res.Add(String.Format("id = {0}, name = {1}", item["id"], item["name"]));
        }
    }
    return res;
}
```

UTILISATION DES DATASETS



Utilisation de ado.net pour l'exécution de requête SQL, ici une insertion :

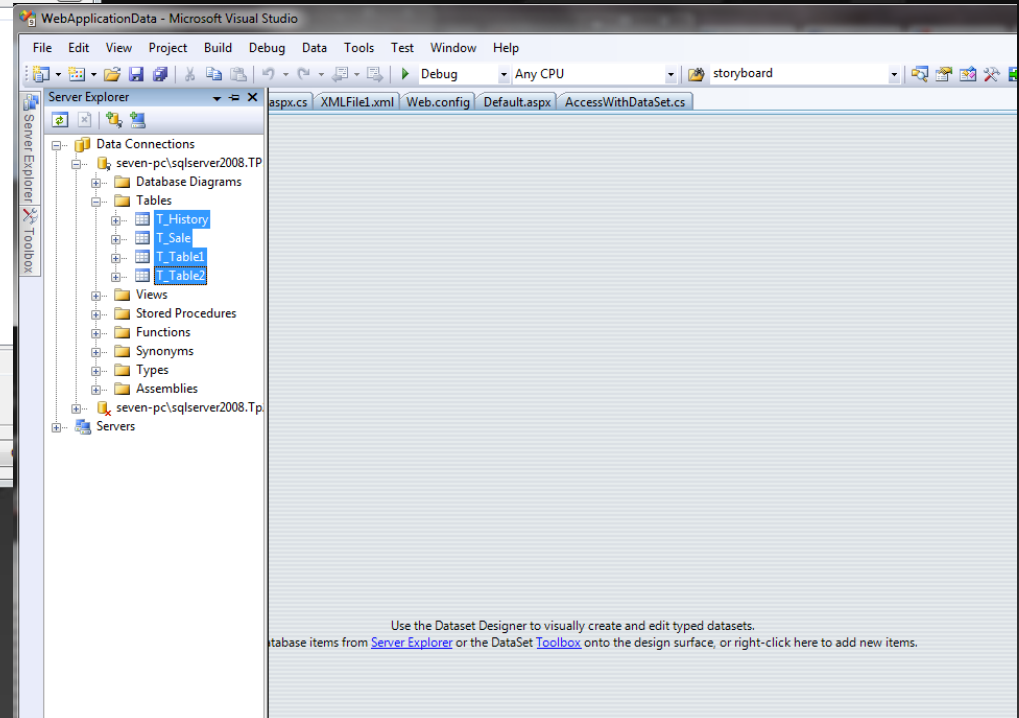
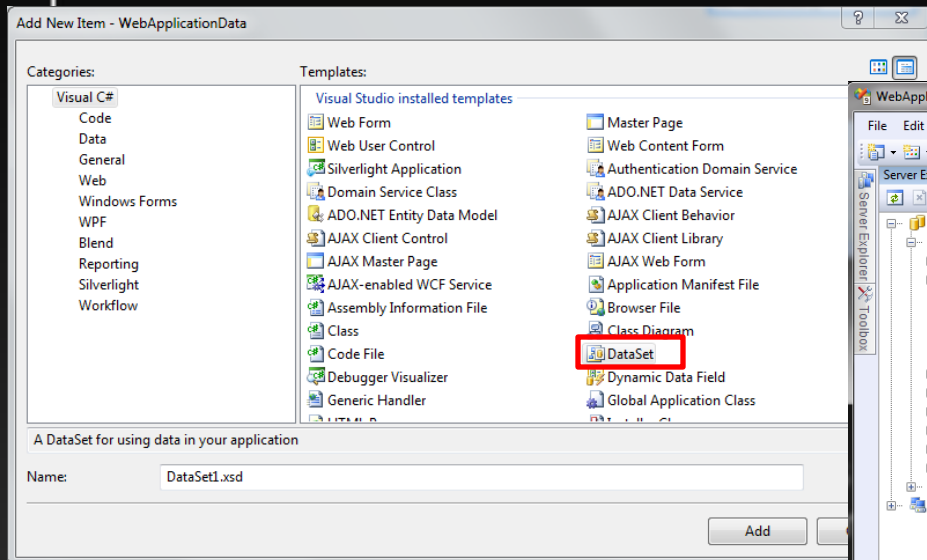
```
public static void insertData()
{
    using (SqlConnection connection = new
        SqlConnection(ConfigurationManager.ConnectionStrings["TPConnectionString"].Connection
            onString))
    {
        System.Data.SqlClient.SqlCommand cmd = new System.Data.SqlClient.SqlCommand();
        cmd.CommandType = System.Data.CommandType.Text;
        cmd.CommandText = "INSERT  dbo.T_Table1 (name) VALUES ('Toshiba')";
        cmd.Connection = connection;

        connection.Open();
        cmd.ExecuteNonQuery();
    }
}
```

UTILISATION DES DATASETS



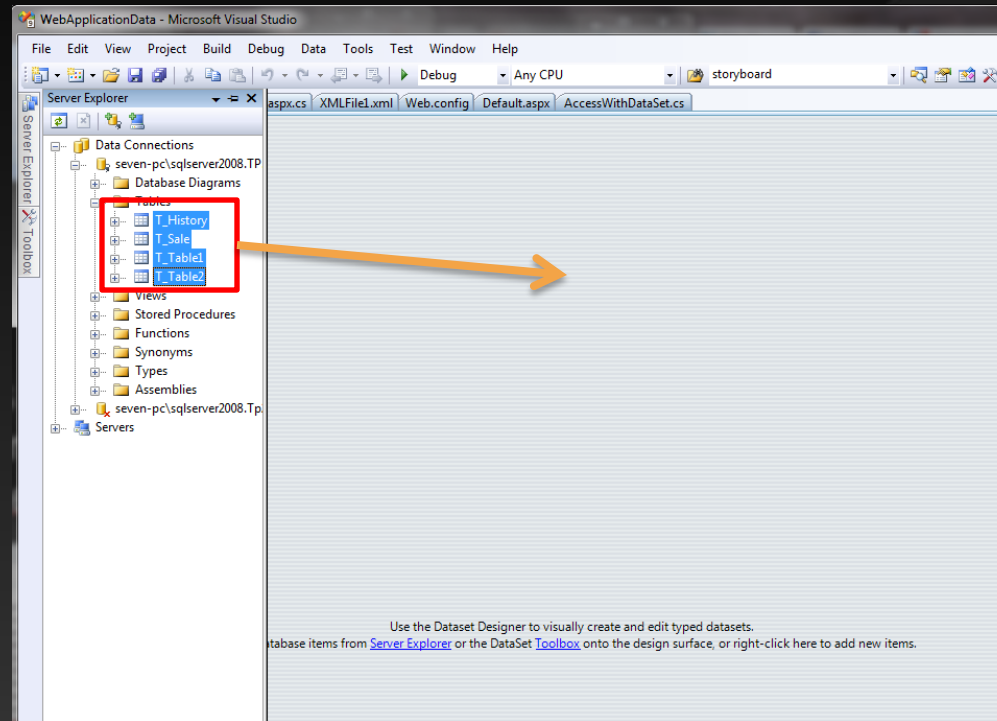
➡ L'utilisation des data set de cette manière n'est pas très productif. Heureusement il existe les datasets typés.



UTILISATION DES DATASETS



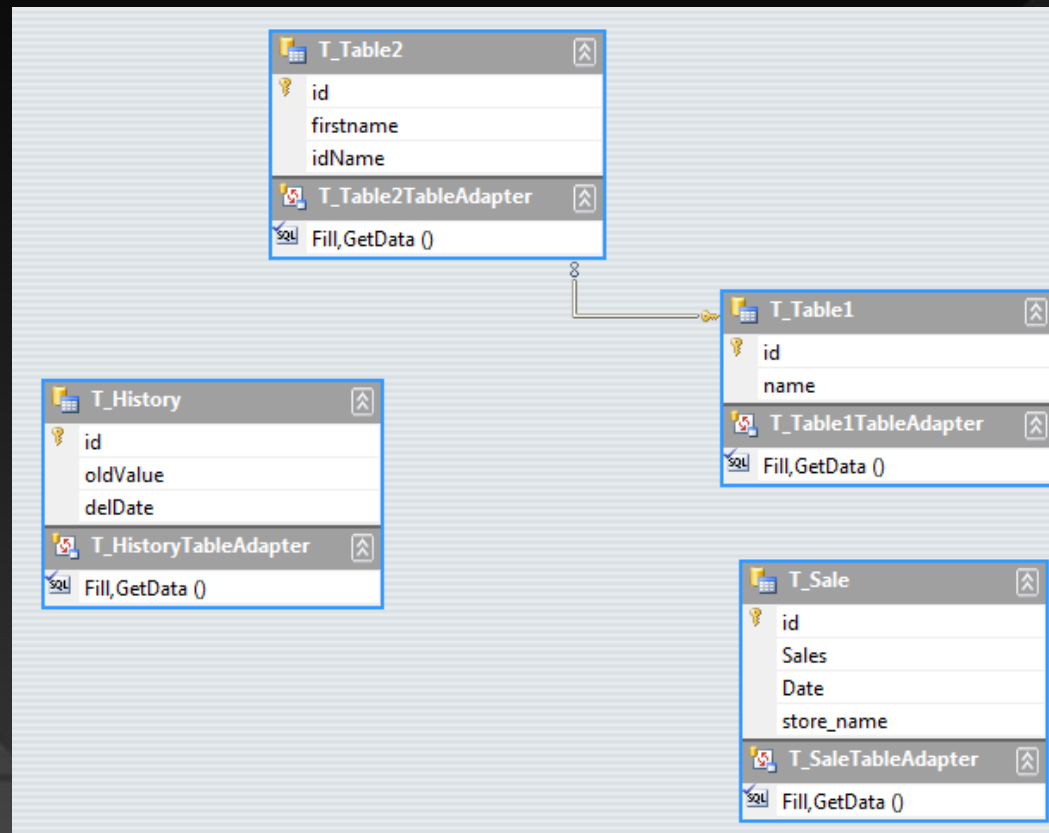
➡ Il ne reste qu'à glisser – déplacer les tables dans la zone grise



UTILISATION DES DATASETS



➡ On obtient le résultat suivant :



UTILISATION DES DATASETS



Le fait d'ajouter et de supprimer via les datasets simplifie la procédure.
Ici la récupération des résultats :

```
public static List<string> GetList()  
{  
    List<string> res = new List<string>();  
  
    DataSet1 dataSet = new DataSet1();  
    DataSet1TableAdapters.T_Table1TableAdapter tableAdapter = new  
    WebApplicationData.DataAccess.DataSet1TableAdapters.T_Table1TableAdapter();  
    tableAdapter.Fill(dataSet.T_Table1);  
  
    foreach (DataRow item in dataSet.T_Table1.Select())  
    {  
        res.Add(String.Format("id = {0}, name = {1}", item["id"], item["name"]));  
    }  
}
```

UTILISATION DES DATASETS



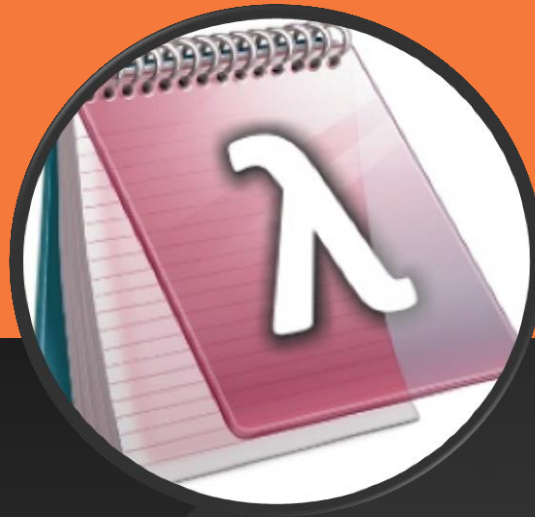
L'insertion :

```
public static void InsertData()  
{  
    DataSet1 dataSet = new DataSet1();  
    DataSet1TableAdapters.T_Table1TableAdapter  
tableAdapter = new  
WebApplicationData.DataAccess.DataSet1TableAdapters.T  
_Table1TableAdapter();  
  
    tableAdapter.Insert("Apple");  
}
```

L'avantage des data sets typés est de permettre un contrôle à la compilation sur les objets de la base, hormis les requêtes qui ne sont pas contrôlées.



LINQ



LINQ

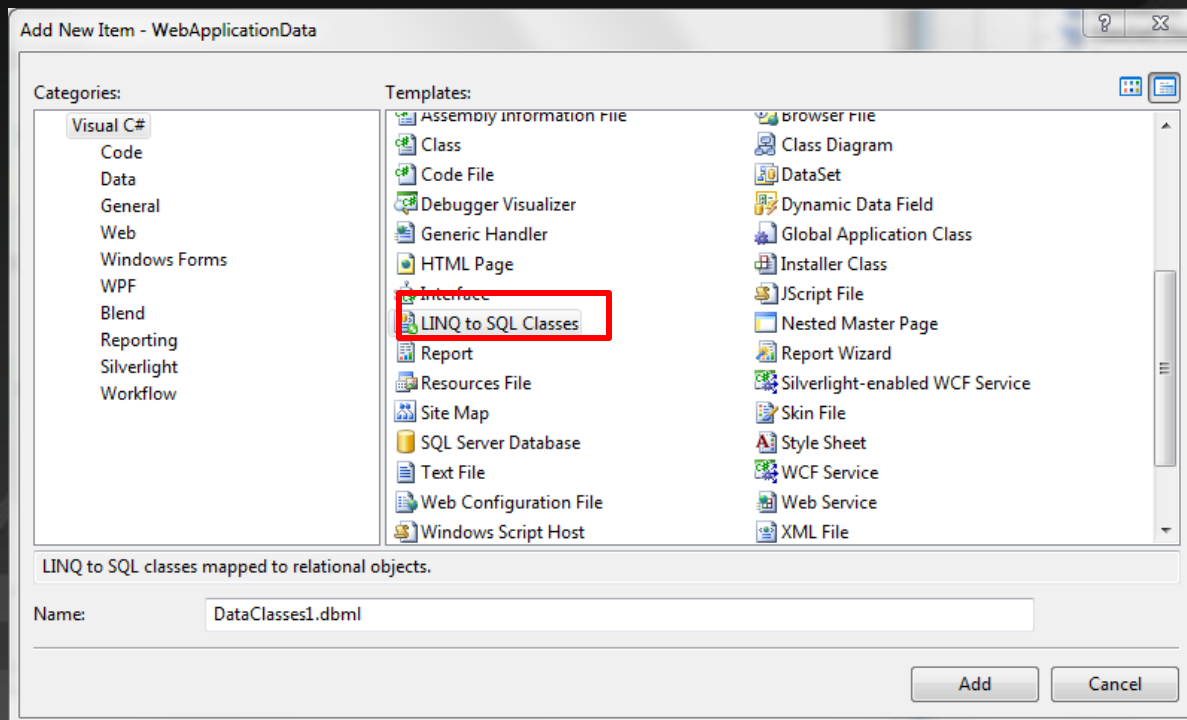


- ➡ Dans le cas où vous avez accès à la version 3.5 de .Net, vous aurez accès à la technologie Linq. Linq signifie Language Integrated Query, et permet de rajouter le contrôle des requêtes à la compilation. Essentiellement couplé pour SQL Server. Cependant différents providers existent pour se mapper avec les bases.

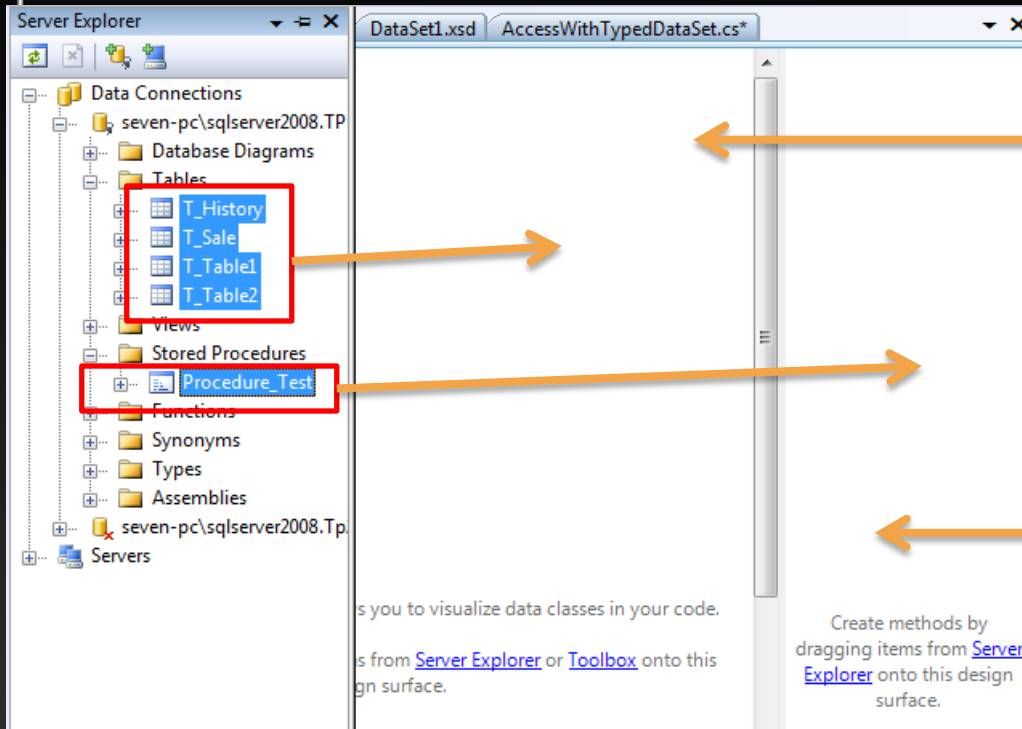
LINQ



- ➡ LinQ n'est pas un ORM, on ne peut que faire un mapping 1 objet => 1 table.
- ➡ Nous allons donc avoir besoin d'ajouter un nouveau fichier au projet un .dbml



LINQ



Permet de mettre les tables de la base de données

Permet de mettre les procédures stockées

LINQ



Utilisation de linq au travers du code :

```
public static List<string> GetList()
{
    List<string> res = new List<string>();
    DataClasses1DataContext dataContext = new DataClasses1DataContext();
    var query = dataContext.T_Table1s.ToList();

    foreach (T_Table1 item in query)
    {
        res.Add(String.Format("id = {0}, name = {1}", item.id,
            item.name));
    }
    return res;
}
```

Une chose importante, pour requêter sur la base, est de construire le contexte. Il est synchronisé à chaque construction.

Autrement dit il n'y a pas de gestion d'accès de concurrences. Les données présentes base correspondront au dernier commit. Les données précédentes seront écrasées

LINQ



Insertion de données :

```
public static void InsertData()  
{  
    DataClasses1DataContext dataContext = new  
    DataClasses1DataContext();  
    T_Table1 table1 = new T_Table1();  
    table1.name = "grandin";  
  
    dataContext.T_Table1s.InsertOnSubmit(table1);  
    dataContext.SubmitChanges();  
}
```

Afin de pouvoir envoyer les données à la base, il ne faut pas oublier d'appeler SubmitChanges(), sinon les données ne seront pas vues des autres.

LINQ



Bon, c'est bien beau... mais, mis à part le fait que tout soit compilé, on a pas encore gagné beaucoup de fonctionnalités. Un des avantages de Linq est également de pouvoir réaliser des requêtes.

```
from nom_var_local in tables_concernées_par_la_requete  
where condition_de_validation  
select resultat_à_retourner
```

Comme on peut le voir la syntaxe reste exactement la même que pour du linq to XML

LINQ



Exemple de requête avec une clause where :

```
public static string GetItem(long id)
{
    DataClasses1DataContext dataContext = new DataClasses1DataContext();
    var query = from tmpItem in dataContext.T_Table1s
                where tmpItem.id == id
                select tmpItem;
    if (query.Any())
    {
        return String.Format("id = {0}, name = {1}", query.First().id,
            query.First().name);
    }
    else
    {
        return String.Empty;
    }
}
```

Pour effectuer une requête, nous devons créer le contexte. On stocke le résultat dans la variable de type VAR (Type générique). On appelle la méthode any() pour déterminer si nous avons eu un résultat, puis on retourne le 1er résultat.

LINQ



Bien entendu, on peut faire des requêtes beaucoup plus complexes et renvoyer un autre type que des strings. Avec linq la bonne pratique est de renvoyer un objet indépendant de linq et donc qui se situe dans la couche DBO. Ceci permettra de transférer les objets à travers WCF.

Requête « complexe » avec une jointure :

```
public static List<string> GetItemJointure()
{
    List<string> res = new List<string>();

    DataClasses1DataContext dataContext = new DataClasses1DataContext();

    var query = from e1 in dataContext.T_Table1s
                join e2 in dataContext.T_Table2s on e1.id equals e2.idName
                select new { e1.name, e2.firstname };

    foreach (var item in query)
    {
        res.Add(String.Format("id = {0}, name = {1}", item.name, item.firstname));
    }
    return res;
}
```


LINQ



Bien que cela reste tout à fait possible de faire de lourdes requêtes avec linq
Il serait plus raisonnable de faire ces requêtes au travers de procédures stockées.
Dans la version 4.0 du framework le code SQL généré s'est vu amélioré. Mais d'un point de vue optimisation , il sera plus simple à un DBA d'améliorer une requête SQL qu'une requête linq.

```
public static List<string> GetItemByProcStock()  
{  
    List<string> res = new List<string>();  
    DataClasses1DataContext dataContext = new DataClasses1DataContext();  
  
    var resRequest = dataContext.Procedure_Test("test");  
    foreach (var item in resRequest)  
    {  
        res.Add(String.Format("nom : {0}", item.name));  
    }  
    return res;  
}
```

LINQ



- ➡ En conclusion, linq permet de gagner énormément de temps au niveau productivité. Cependant, il faut faire attention lors de lourds traitements à faire travailler la base de données.
- ➡ De plus, la version de LinQ to SQL n'évoluera plus, Microsoft se concentrant sur Entity Framework

LINQ



➡ Quelques outils utiles :

- ▶ <http://www.linqpad.net/>
 - ▶ Permet de visualiser les requêtes linQ en SQL
 - ▶ D'interroger des bases de données en Linq
 - ▶ Support d'Entity Framework, OData, ...



LES MÉTHODES D'EXTENSION

LES MÉTHODES D'EXTENSION



Ce mécanisme est apparu avec le framework 3.5. Cela permet de rajouter des méthodes à des types, sans avoir besoin de les modifier ou de les recompiler.

Exemple d'une extension de méthode :

```
namespace WebApplicationData.ExtensionsMethods
{
    public static class ExtensionString
    {
        public static int WordCount(this string chaine)
        {
            return chaine.Split(new char[] { ' ', '.', '?' }).Length;
        }
    }
}
```

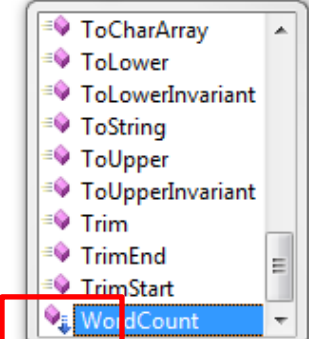
Chaque méthode doit être statique et avoir le mot clé « this » dans ses paramètres

LES MÉTHODES D'EXTENSION



Ensuite, pour pouvoir l'utiliser il suffit de faire un using sur la namespace de la méthode.

```
protected void Page_Load(object sender, EventArgs e)
{
    string strTest = "ceci est un test";
    int cpt = strTest.Wo|
}
```



Symbole d'extension de
méthodes

```
using WebApplicationData.ExtensionsMethods;

namespace WebApplicationData
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            string strTest = "ceci est un test";
            int cpt = strTest.WordCount();
        }
    }
}
```

LES MÉTHODES D'EXTENSION



- ➡ L'utilisation d'extension de méthodes, permet de simplifier la programmation mais aussi de rendre plus lisible le code source, comme par exemple lors de manipulation de dates.



POINT ARCHITECTURAL



POINT ARCHITECTURAL



- ➡ Maintenant que nous avons avancé un peu plus dans le programme, il faut recentrer un peu les idées.

DBO

INTERFACE (WEBSERVICE, ASP.NET)

BUSINESS MANAGEMENT (EXTENSIONS METHODS)

Linq To Xml

Txt

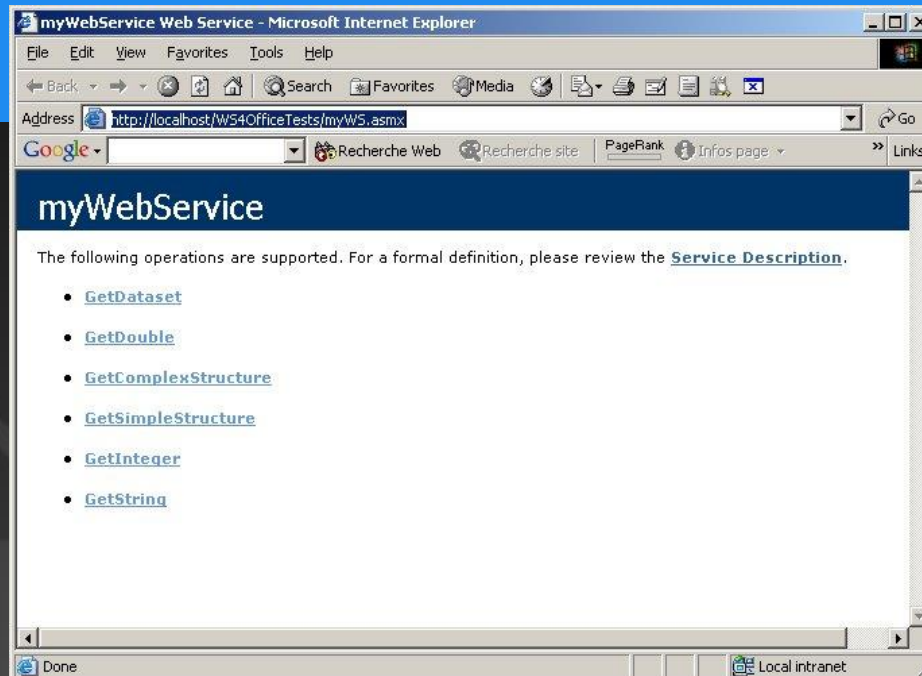
Entity Framework

DataSet

Linq To SQL



WEB SERVICE



WEB SERVICE

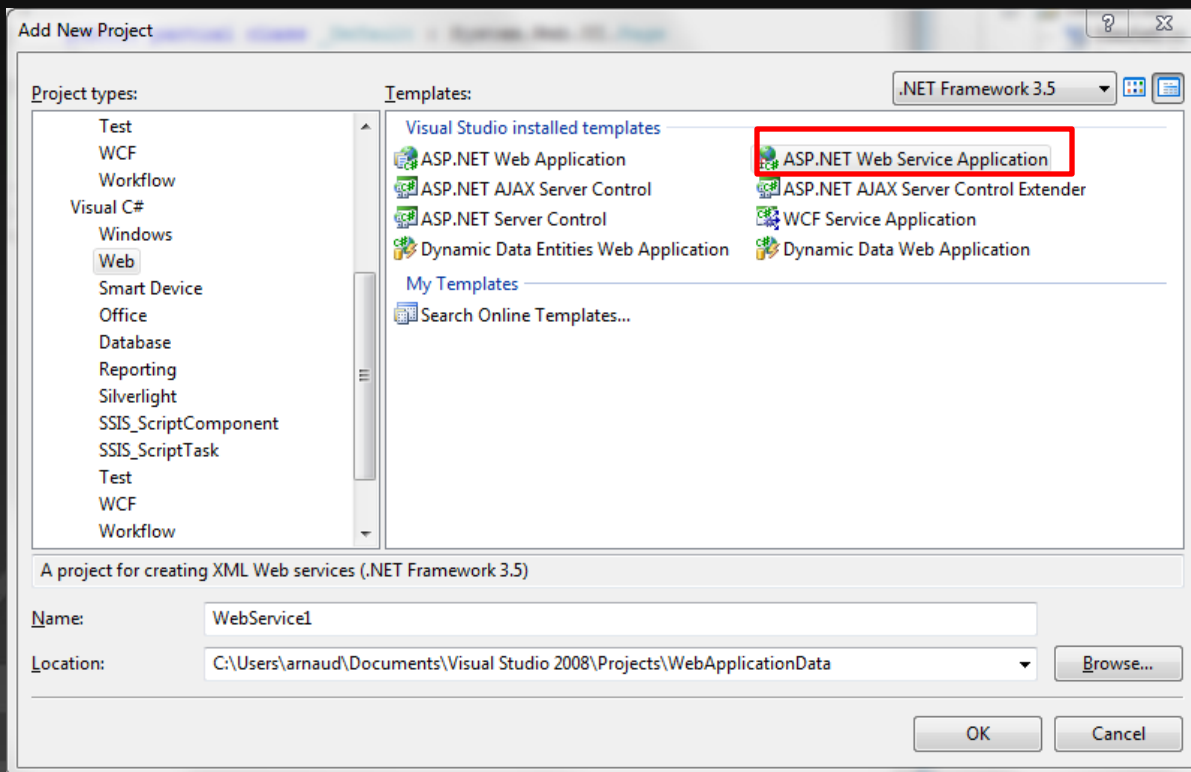


- ➡ Avec le framework, il existe plusieurs moyens de faire des web services. Soit avec WCF, soit avec les anciennes méthodes, autrement dit sous forme d'asmx.

WEB SERVICE



- ➡ Pour y parvenir, il faut créer un nouveau projet de type asp.net web service application.



WEB SERVICE



➡ Ensuite il ne reste plus qu'à écrire les méthodes.

```
using System.Web.Services;

namespace WebServiceTP
{
    /// <summary>
    /// Summary description for Service1
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    // To allow this Web Service to be called from script, using
    // [System.Web.Script.Services.ScriptService]
    public class Service1 : System.Web.Services.WebService
    {
        [WebMethod]
        public string HelloWorld()
        {
            return "Hello World";
        }
    }
}
```

WEB SERVICE



- ➡ Pour exposer les méthodes, il suffit juste de rajouter l'attribut `[WebMethod]` sur la fonction.



QUESTIONS ?