Services réseaux

TP n°1 : Introduction

Antony BRUGERE
anto.brugere@gmail.com

Frédéric CHASSAGNE
frederic.chassagne@capgemini.com

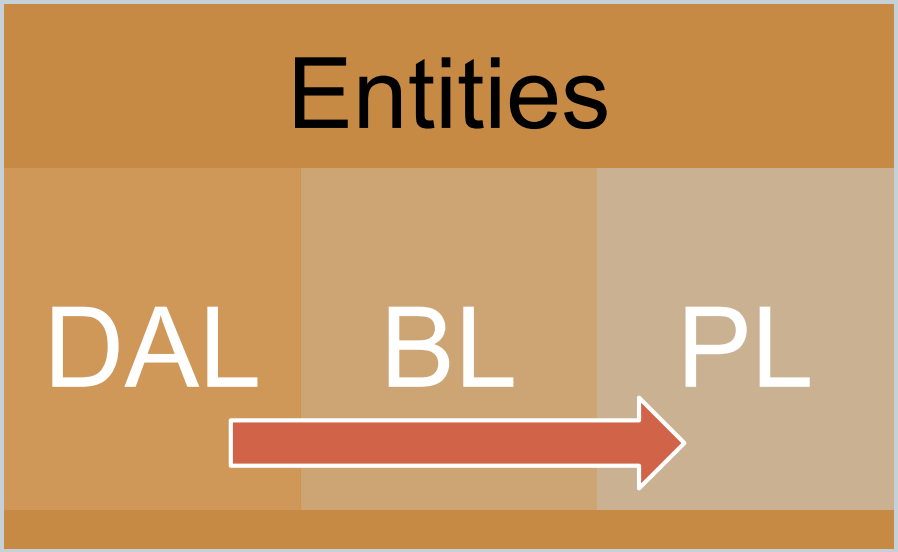
Plan du TP intégré

- Architecture 4 tiers
  - Création projet
- Rappel syntaxe C# et interfaces
  - Implémentation
- Les collections
  - Implémentation
- Syntaxe Linq

Architecture logicielle

- Couches du modèle 4 tiers
  - Presentation Layer
    - Interface Homme Machine
  - Business Layer
    - Noyau métier
  - Data Access Layer
    - Accès aux données
  - Entities Layer
    - Objets basiques

    ie Plain Old Clr Objects



Etape 1

- Création solution et projets
- Où vont se trouver les différentes classes ?

Syntaxe C#

- Langage Case sensitive
  - Bonjour != BonJour
- Convention de nommage
  - Méthodes notées MaMethode()
  - Variables notées maVariable
  - Variables de classe notées _maVariable
  - Pas de notation hongroise (sauf ihm)

Syntaxe C# - Enum

- Enum = sous-ensemble de valeurs prédéfinies
- enum jour { lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche}
- par défaut : rang de lundi=0, rang de mardi=1, … , rang de dimanche=6
- Multiplexage possible

Syntaxe C# - Struct

```
struct Point
{
int x;
Int y;
void switch(){
        int tmp = y;
         y = x; x = tmp;
    }
}
```

Syntaxe C# - Transtypage

- Transtyper = changer une variable de type
- Transtypage implicite
  - Int n = 12; float f = n;
- Transtypage explicite
  - Float f = 12f; n = (int) f;
- Conversion > Transtypage
  - Convert.To…(obj)

Syntaxe C# - Opérateurs

- Arithmétiques
  - + - * /
  - % : Reste
- Incrémentation
  - Pré-incrémentation : ++var
  - Post-incrémentation : var++

Syntaxe C# - Opérateurs

- Logiques
  - ! : non
  - & : et
  - | : ou
  - && : et courcircuité
  - || : ou courcircuité

Syntaxe C# - Choix

- If
  - Forme :

    if (cond) { traitement si cond vérifié }

    else { traitement sinon }
  - Version condensée :
    - Expression ? Valeur : valeur;
- Switch (cond)

  { case 1 : … ; break;

  default : …; break;

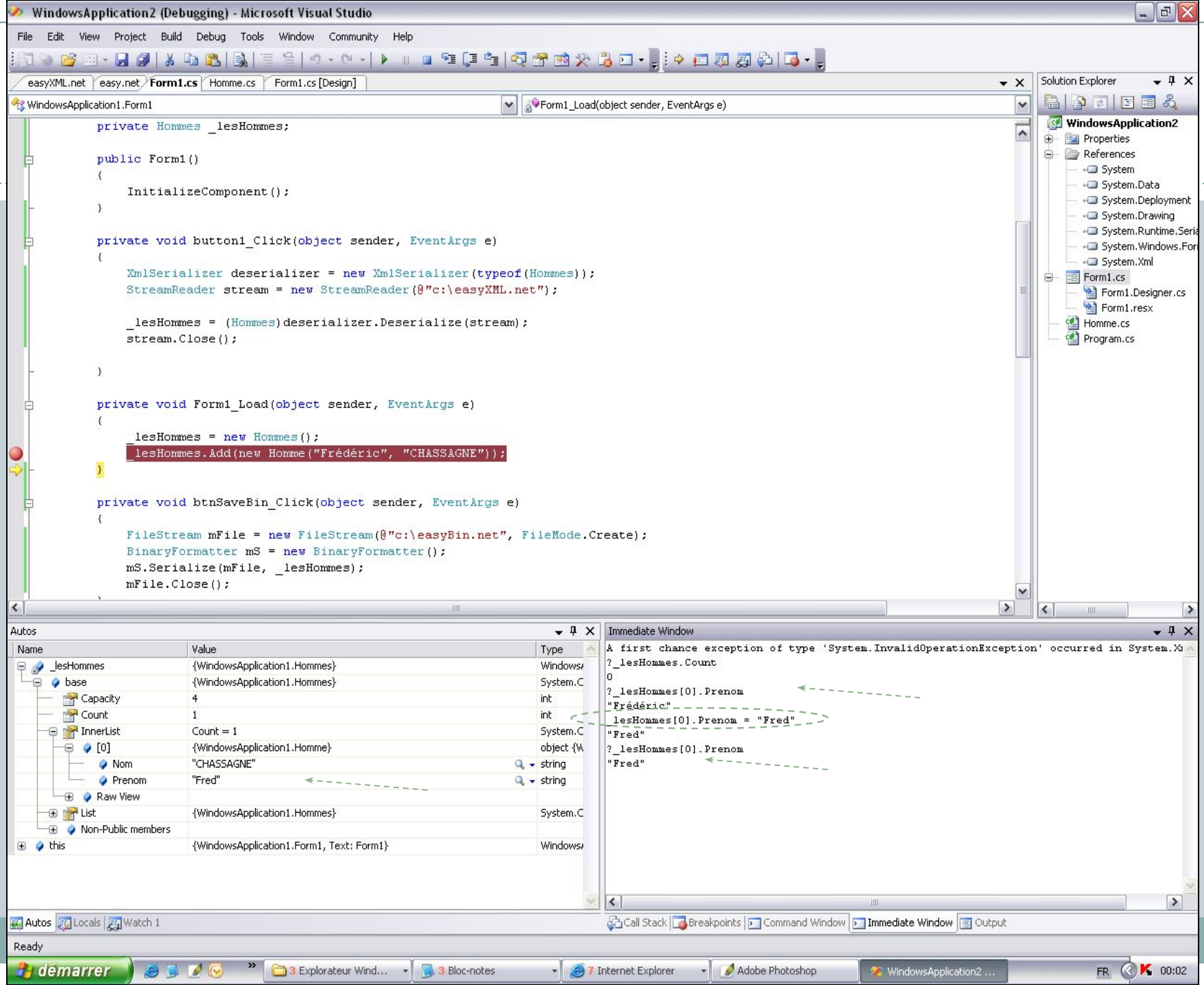
  }

Syntaxe C# - Boucles

- While (cond) { traitement}
- Do { traitement } while (cond)
- For(int i = 0; i< 10, i++){ traitement }
- Foreach(int i in tab){traitement}

C# - Objet Console

- Console.Readline();
  - Lecture de paramètre
- Console.WriteLine();
  - Affichage sur la sortie standard
- Directives de compilation
  - #DEBUG


WindowsApplication2 (Debugging) - Microsoft Visual Studio
File Edit View Project Build Debug Tools Window Community Help
easyXML.net easy.net Form1.cs Homme.cs Form1.cs [Design]
WindowsApplication1.Form1
Form1_Load(object sender, EventArgs e)
private Hommes _lesHommes;
public Form1()
{
InitializeComponent();
}
private void button1_Click(object sender, EventArgs e)
{
XmlSerializer deserializer = new XmlSerializer(typeof(Hommes));
StreamReader stream = new StreamReader(@"c:\easyXML.net");
_lesHommes = (Hommes)deserializer.Deserialize(stream);
stream.Close();
}
private void Form1_Load(object sender, EventArgs e)
{
_lesHommes = new Hommes();
_lesHommes.Add(new Homme("Frédéric", "CHASSAGNE"));
}
private void btnSaveBin_Click(object sender, EventArgs e)
{
FileStream mFile = new FileStream(@"c:\easyBin.net", FileMode.Create);
BinaryFormatter mS = new BinaryFormatter();
mS.Serialize(mFile, _lesHommes);
mFile.Close();
Solution Explorer
WindowsApplication2
Properties
References
System
System.Data
System.Deployment
System.Drawing
System.Runtime.Seria
System.Windows.For
System.Xml
Form1.cs
Form1.Designer.cs
Form1.resx
Homme.cs
Program.cs
Autos
Name Value Type
_lesHommes {WindowsApplication1.Hommes} Windows
base {WindowsApplication1.Hommes} System.C
Capacity 4 int
Count 1 int
InnerList Count = 1 System.C
[0] {WindowsApplication1.Homme} object {W
Nom "CHASSAGNE" string
Prenom "Fred" string
Raw View
List {WindowsApplication1.Hommes} System.C
Non-Public members
this {WindowsApplication1.Form1, Text: Form1} Windows
Immediate Window
A first chance exception of type 'System.InvalidOperationException' occurred in System.X
?_lesHommes.Count
0
?_lesHommes[0].Prenom
"Frédéric"
_lesHommes[0].Prenom = "Fred"
"Fred"
?_lesHommes[0].Prenom
"Fred"
Autos Locals Watch 1
Call Stack Breakpoints Command Window Immediate Window Output
Ready
démarrer
3 Explorateur Wind...
3 Bloc-notes
7 Internet Explorer
Adobe Photoshop
WindowsApplication2 ...
FR
00:02

Environnment Visual Studio

- Mode Debug
  - Breakpoints
    - Ajout / Suppression par F9
  - Debug Pas à Pas
    - Step Into (F11)
    - Step Over (F10)
  - Immediate Window

Syntaxe C# - Propriétés

- Accesseurs à une variable
- Syntaxe

```
public bool MaPropriete
    {
        get
        {
            return _maPropriete;
        }
        set
        {
            _maPropriete= value;
        }
    }
```

Syntaxe C# - Interfaces

- Aucune implémentation
- Ensemble de méthodes que l'objet doit suivre

= "contrat de service" du composant

- Code réalisé dans les classes qui l'implémente

- BL communique via des interfaces
  - Sécurité
  - Modularité

Syntaxe C# - Classes abstraites

- Classe incomplète, non instanciable
- Contient 1 ou plusieurs méthodes abstraites
- Code réalisé dans les classes qui l'hérite

- Entities contient des classes abstraites
  - Possibilité de manipuler des ensembles
  - Centralise la gestion des attributs communs

Etape 2

- Création des objets de la couche Entities

C# - Les collections

IEnumerable

ICollection

IList

C# - Les collections

- Namespace : System.Collections.Generics
- Containers génériques
  - Avantage : Contient des objets typés dont le type est précisé dans l'instanciation
- Collections existantes
  - List <Class>
  - Stack <Class>
  - Queue<Class>
  - Dictionary<keyClass, valueClass>

C# - List<Class>

- Implémente IList
- Non dimensionné
- Générique
- Gestion
  - Ajout : Add(<Class> o);
  - Suppression : Remove(<Class> o); RemoveAt(position);
  - Accès : MaListe[position];
  - Taille : Count();

C# - Stack<Class>

- Implémente ICollection, IEnumerable
- Concept : LIFO
- Gestion
  - Ajout : Push(<Class> o);
  - Suppression : Pop();
  - Accès : Peek();
  - Taille : Count();

C# - Queue<Class>

- Implémente ICollection, IEnumerable
- Concept : FIFO
- Gestion
  - Ajout : Enqueue(<Class> o);
  - Suppression : Dequeue();
  - Accès : Peek();
  - Taille : Count();

C# - Dictionary<keyClass, valueClass>

- Implémente ICollection, IEnumerable
- Clés – valeurs
- Gestion
  - Ajout : Add(Tkey key, Tvalue value);
  - Suppression : Remove(Tkey key);
  - Accès : Hash[key];
    - Warning l'accès retourne une exception si la clef n'existe pas.
  - Taille : Count();

Etape 3 : Couche business

- Création du manager
  - Quelles méthodes vont être présentes ?
  - Constructeur
  - Création de la liste exemple

Syntaxe Linq

- Language-Integrated Query, ou Requêtage integré au langage = Un ajout marquant du Framework 3.5 et de C# 3.0
- Linq permet
  - L'interrogation uniforme quelque soit le type de données
  - Récupération et manipulation de données
  - Couche d'abstraction des données

Syntaxe Linq

- Linq se décompose en
  - Linq To Objects : manipulation des collections en .net
  - Linq To ADO.Net (*ie* Linq To SQL, Linq To DataSet et Linq To Entities) : Récupération et manipulation des données d'un SGBD
  - Linq To XML : Récupération et manipulation des données d'un modèle xml

Syntaxe Linq

- Notre scope : Linq To Objects
- But :
  - Manipulation des collections d'objets
  - Parcours, tri, filtres sur les collections
  - Remplacement des boucles while, foreach, …

Syntaxe Linq

- Exemple de requête Linq :

```
IEnumerable<string> results = from x in auteurs
                              where x.Prenom.StartsWith("Ai")
                              select x.Prenom + ' ' + x.Nom;

                    OU

Ienumerable<string> results = auteurs.Where(x = > x.Prenom.StartsWith
    (« Ai »)).Select(x => x.Prenom + ' ' + x.Nom);
```

Syntaxe Linq

- Possibilités d'ajouter à une requête linq
  - Fonction de calcul : .Count(), .Max(), …
  - Notion d'agrégat : .Intersect(), .Except(), …

Etape 4 : Couche de présentation

- Implémentation du menu console
- Utilisation de Linq dans le manager