TP n°1 : Introduction

SERVICES RÉSEAUX

Hélène CHASSAGNE
helene.chassagne@orange.fr

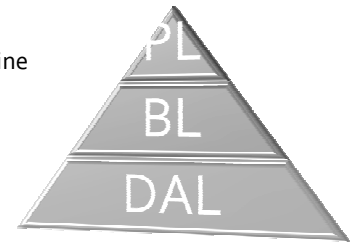
Frédéric CHASSAGNE
frederic.chassagne@atosorigin.com

Plan du TP intégré

- Architecture 3 tiers
  - Création projet
- Rappel syntaxe C# et interfaces
  - Implémentation
- Les collections
  - Implémentation

Architecture 3 tiers

- Couches du modèle
  - Presentation Layer
    - Interface Homme Machine
  - Business Layer
    - Noyau métier
  - Data Access Layer
    - Accès aux données



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;

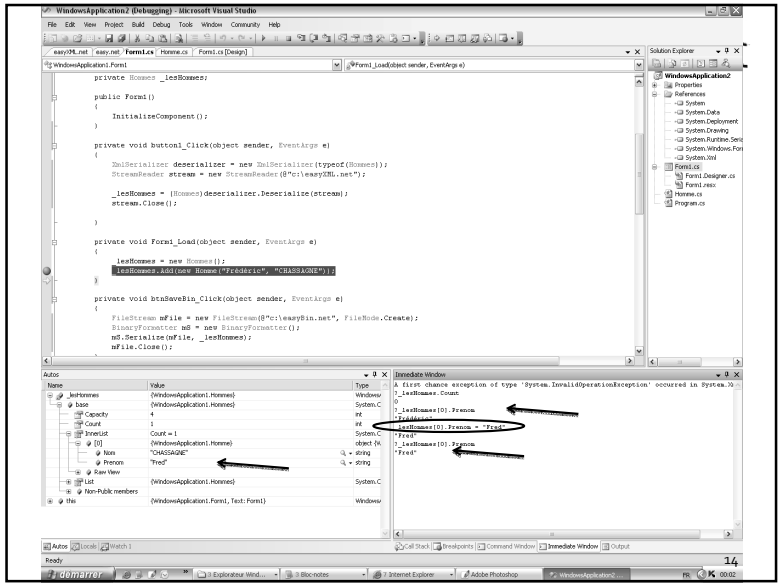
  }

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



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

- BL communique via des interfaces
  - Sécurité
  - Modularité

Etape 2 : Couche business

- Création des classes Livre et classes associées
  - Interface d'accès
  - Attributs
  - Accesseurs
  - Constructeur

C# - Les collections

IEnumerable

⇩

ICollection

⇩

IList

C# - Les collections

- Namespace : System.Collections
- Collections existantes
  - ArrayList
  - Stack
  - Queue
  - Hashtable

C# - ArrayList

- Implémente IList
- Non dimensionné
- Non typé
- Gestion
  - Ajout : Add(Object o);
  - Suppression : Remove(Object o); RemoveAt(position);
  - Accès : MaListe[position];
  - Taille : Count();

C# - Stack

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

C# - Queue

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

C# - Hashtable

- Implémente ICollection, IEnumerable
- Clés – valeurs (type object)
- Gestion
  - Ajout : Add(Tkey key, Tvalue value);
  - Suppression : Remove(Tkey key);
  - Accès : Hash[key];
  - Taille : Count();

Génériques

- Version générique des collections
  - System.collections.generics
  - Pouvoir préciser le type

  - ArrayList → List <Class>
  - Stack → Stack <Class>
  - Queue → Queue<Class>
  - Hashtable → Dictionary<keyClass, valueClass>
    - !!! Dic[key] exception si key n'existe pas, alors que Hash[key] renvoie null

Etape 3 : Couche de présentation

- Classe Auteur avec collection
- Gestion du panier
- Gestion de la bibliothèque
- Implémentation du menu console