TP n°2 : .net – Smart appli

SERVICES RÉSEAUX

Hélène CHASSAGNE
helene.chassagne@orange.fr

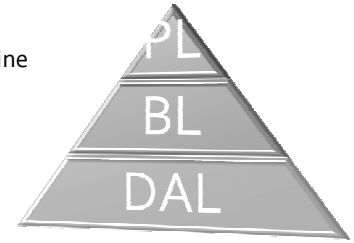
Frédéric CHASSAGNE
frederic.chassagne@atosorigin.com

Plan du tp intégré

- Résumé du TP précédent
- IHM « Smart client »
- UserControl
- Events

Architecture 3 tiers

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



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 1 : Mise en forme TP1

- Dialogue entre couches via des Interfaces
- Commentaires et 1 fichier par classe C#
- Gérer des types Ienumerable<ILivre> dans la couche Business
- Gérer les types de livre sous forme Enum

Plan du tp intégré

- Résumé du TP précédent
- IHM « Smart client »
- UserControl
- Events

IHM « Smart Client »

- Windows.Forms = IHM « Smart Client »
- Avantages
  - Accès complet à l'ordinateur & au réseau
  - Aucun maintien de session
- Inconvénients
  - Peu interopérable
  - Installation nécessaire

Etape 2 : Création du projet

- Créer un projet de type Windows Form
- Visualiser la boite des contrôles

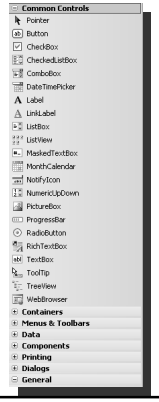
IHM « Smart Client »

- Containers = contrôle contenant des contrôles
- Rôle
  - Faciliter les regroupements par groupes
    - Ex : GroupBox
  - Simplifier la gestion de la conception
  - Simplifier la gestion du redimensionnement
    - Granularité déplacée
    - Gestion en cascade

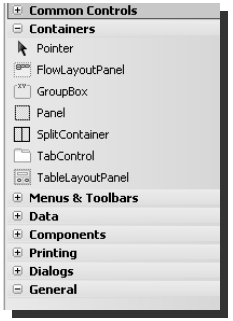
IHM « Smart Client »

- Control = Brique de base de l'IHM
- Rôle
  - Permettre la communication avec le programme
  - Simplifier la vie de l'utilisateur
- Idée générique dans .net
  - Code haut niveau

IHM « Smart Client »



IHM « Smart Client »



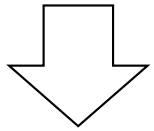
IHM « Smart Client »

- Containers + Contrôles = IHM
- Pas d'intelligence métier
- Interaction utilisateur

- Brique la plus importante de l'application !

IHM « Smart Client »

- But d'un programme = satisfaction client
- Satisfaction client = jolie interface



- But d'un programme = jolie interface

IHM « Smart Client »

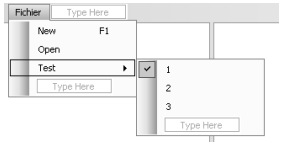
- Points importants d'une belle interface
  - Regroupements logique
  - Alignement des contrôles
  - Pas de formulaire « placard à tout faire »
  - Mise en avant des informations importantes
  - Intéraction avec l'environnement
    - Sauvegarde des dimensions / positions de l'ihm
    - Gestion des erreurs « Friendly »
  - Privilégier le « Windows » look

IHM « Smart Client »

- Composants avancés
  - MenuStrip
  - ContextMenuStrip
  - Treeview
  - Listview
  - SplitContainer

IHM « Smart Client »

- MenuStrip
  - But : Gérer le menu de l'application



- Alternative : Menu

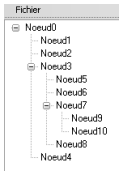
IHM « Smart Client »

- ContextMenuStrip
  - But : Proposer un accès rapide à certaines fonctionnalités via un clic droit

- Inscription nécessaire des composants souhaitant utiliser le ContextMenuStrip
- Alternative : ContextMenu

IHM « Smart Client »

- Treeview
  - But : Présenter des informations hiérarchique sous forme d'arborescence

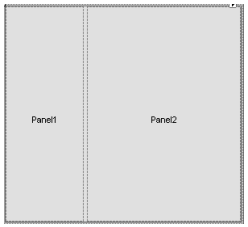


IHM « Smart Client »

- Listview
  - But : Présentation d'objet sous differentes formes
- 4 formes possibles :
  - Icon (large & small) (à combiner avec ImageList)
  - Detail
  - List
  - Tile

IHM « Smart Client »

- SplitContainer
  - But : Afficher 2 containers séparés par un splitter (ie barre de redimensionnement horizontal ou vertical)



IHM « Smart Client »

- Objectifs d'une IHM
  - Redimensionnable à souhait
  - Mémoire de position, grandeur
  - Intégrée à l'OS sur lequel elle tourne

Etape 2 : Couche de présentation

- Gestion du panier
- Gestion de la bibliothèque
- → Implémentation Formulaire principal

Plan du tp intégré

- Résumé du TP précédent
- IHM « Smart client »
- UserControl
- Events

UserControl

- Contrôles personnalisés
  - Disposition des contrôles tels un formulaire
  - Gestion des évènements associés
  - Réutilisable à souhait
- 1 contrôle personnalisé = ∑ contrôles présentés & utilisés comme une brique unitaire

UserControl

- Propriétés visibles dans Visual Studio
- Attributs
  - Category : Sous-ensemble où placer la propriété
  - Description : Description relative à la propriété
  - Browsable : Visibilité dans l'onglet de propriétés

```
[Category("Configuration"), Browsable(true), Description("Saisir le titre à
   afficher")]
public String Title
{   get   { return this.txtTitre.Text;  }
   set   { this.txtTitre.Text = value;  }
}
```

Plan du tp intégré

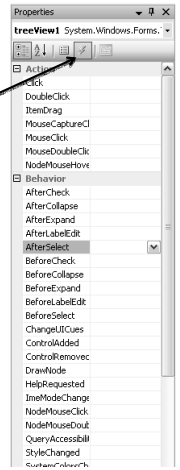
- Résumé du TP précédent
- IHM « Smart client »
- UserControl
- Events

Events

- Event : Code déclenché par une action de l'utilisateur
  - Exemple : Clic() sur un bouton
- Déclaration en C#
  - this.button1.Click += new System.EventHandler(this.button1_Click);
- Délégué & event
  - Mécanisme d'abonnement
  - Exécution synchrone

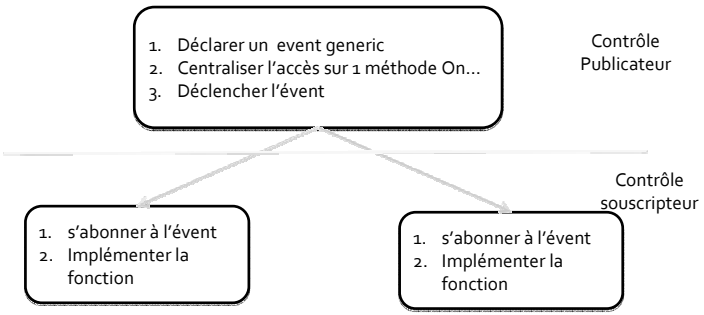
Events

- VS.Net
  - Panneau d'accès
  - Ajout automatique



Les events

- Mécanisme



Les events

- Exemple : Ajout d'un event signalant un ajout de livre

1. Déclarer, si besoin, un MonEventArg héritant de EventArgs

```
public class LivreAddedEventArgs :
EventArgs
  {
    private ILivre _livre;

    public ILivre Livre
    {
      get { return _livre; }
    }

    public LivreAddedEventArgs(ILivre livre)
    {
      _livre = livre;
    }
  }
```


Les events

2. Déclarer l'event generic

```
public event EventHandler<LivreAddedEventArgs> LivreAdded;
```

3. Centraliser l'accès

```
protected virtual void OnLivreAdded(LivreAddedEventArgs e)
{
  if (LivreAdded != null) LivreAdded(this, e);
}
```

4. Déclencher l'évent dans le code

```
private void btnAjoutPanier_Click(object sender, EventArgs e)
{
  OnLivreAdded(new LivreAddedEventArgs(_innerLivre));
}
```