heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Analyse et programmation 2

Introduction à .Net et au langage C#

heig-vd

Thèmes abordés

- La plateforme .Net
  - Architecture.
  - Code MSIL, compilation à la volée.
  - Multi-langage.
  - Gestion de la mémoire.
  - Performances
- Le langage C#
  - Similitudes et différences avec C++.
- Développement visuel en C# avec Visual Studio

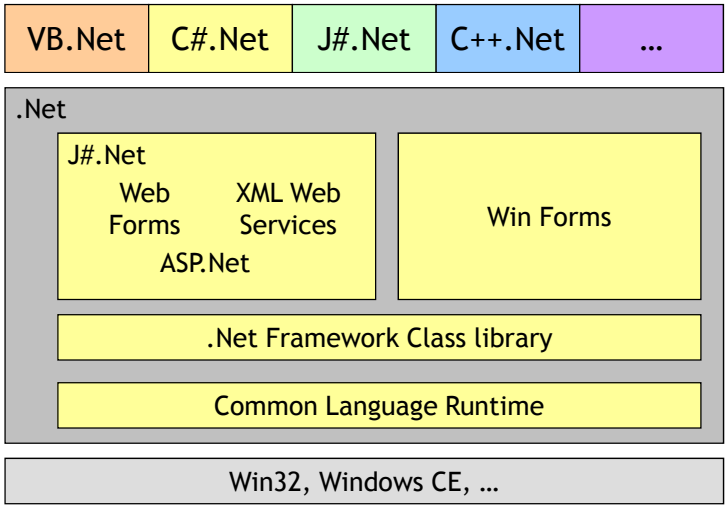
La plateforme .Net

Pour la petite histoire…

- Anders Hejlsberg
  - Concepteur de Turbo Pascal et de Delphi.
  - A l'origine des concepts qui ont fait le succès de ces outils.
  - Débauché par Microsoft en 1996.
  - Concepteur de la plateforme .Net.
  - On retrouve dans .Net les concepts qui donnent à Delphi sa puissance expressive, sous une forme encore améliorée.
  - Nombreuses similarités entre les concepts .Net et Delphi.

La plateforme .Net.

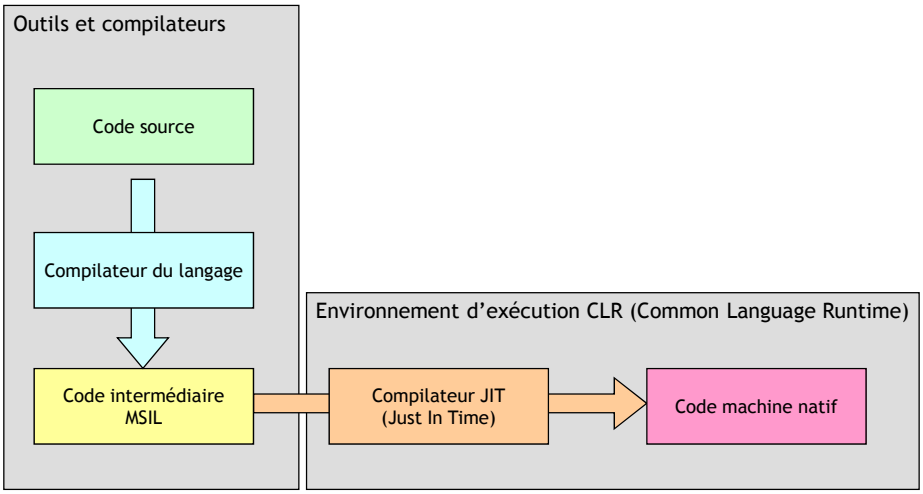
Architecture – multi langage et multi plateforme



heig-vd

La plateforme .Net.

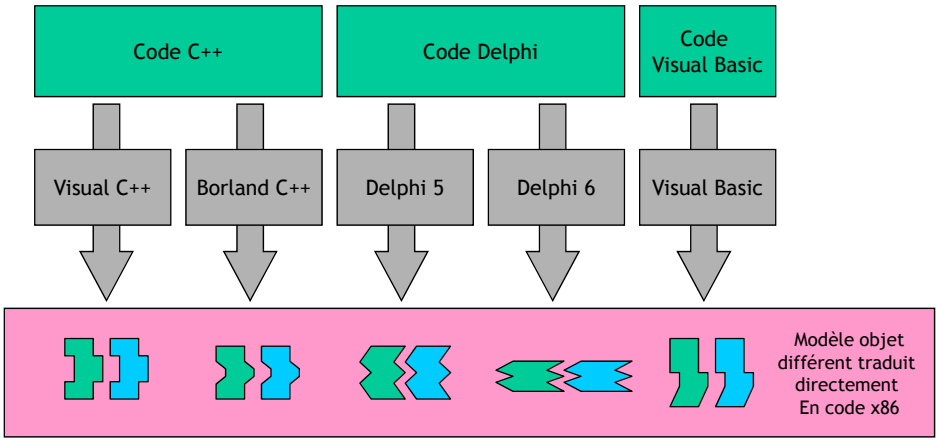
Génération de code .Net



heig-vd

La plateforme .Net.

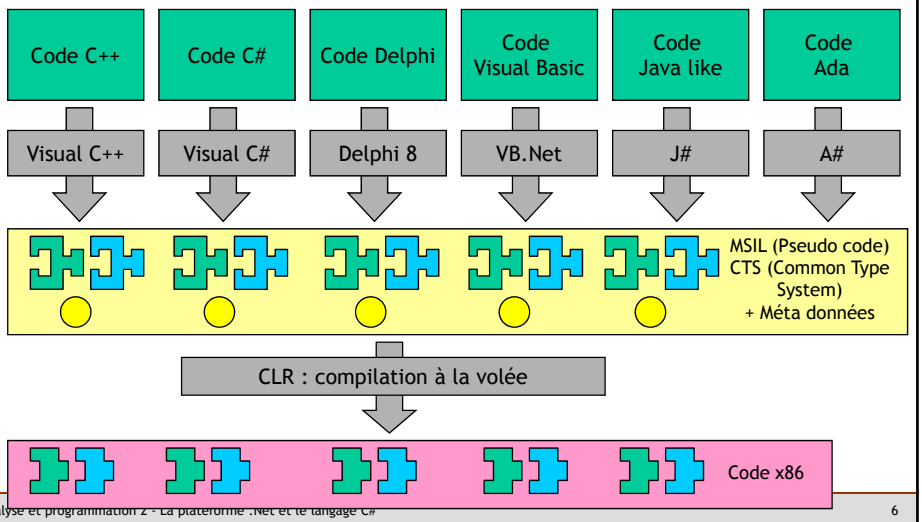
Avant: génération de code machine



heig-vd

La plateforme .Net.

Génération de code .Net



heig-vd

La plateforme .Net

Conséquences de cette approche

- Interopérabilité
  - Tous les langages génèrent un pseudo code basé sur le même modèle objet.
  - Des appels entre langages différents sont supportés sans coût.
  - Les méta données permettent à tous les compilateurs d'accéder aux éléments contenus dans un assemblage compilé.
- Sécurité
  - Le CLR peut interdire le code non sûr s'il provient d'Internet.
  - Permet de supprimer de nombreux types d'attaques virales.
- Portabilité
  - Le code produit n'est pas spécifique à un micro processeur.
  - .Net allégé pour l'informatique embarquée: .Net Compact Framework.
  - Portage sur Linux partiellement opérationnel (Mono).
- Non déterminisme
  - La compilation à la volée induit des temps d'exécution non prévisibles.

heig-vd

La plateforme .Net
Gestion de la mémoire .Net

- Seuls les objets alloués peuvent être accédés.
  - Plus de risque de corruption de mémoire difficile à localiser.
  - Empêche un grand nombre d'attaques virales.
- Comptage de référence
  - Le nombre de références sur un objet est automatiquement géré par le système.
  - Un objet est libéré par le CLR lorsqu'il n'est plus référencé. Pas de risque d'oubli de libération de mémoire.
- Allocation et libération des blocs de mémoire par le CLR.
  - Gestion de la mémoire sous contrôle du CLR.
  - Aucune attention requise du programmeur.
  - Ramasse miettes : non déterminisme.

heig-vd

La plateforme .Net
Gestion de la mémoire .Net

- Illustration

```
int[] tableau;
tableau = new int[100];
delete tableau;
```

- Plus de libération explicite
  - L'opérateur delete n'existe pas !
  - La mémoire est libérée automatiquement.
  - Le moment de la libération est défini par le CLR.
  - GC: Garbage Collection.

heig-vd

Le langage C#
Aperçu de la syntaxe

- Principe
  - Syntaxe proche du C++, mais allégée.
  - Instructions et types de données similaires au C++.
- Différences essentielles
  - Langage orienté objet pur.
    - Il n'est pas possible de créer une fonction !
    - Toute fonction est nécessairement rattachée à une classe.
    - Les fonctions appelables sans créer d'objet doivent être déclarées statiques.
  - Classes de visibilité
    - La classe de visibilité doit être rappelée à chaque déclaration.
  - Fichiers en-tête
    - Il n'y a plus lieu de créer un fichier en-tête séparé.

heig-vd

Le langage C#
Aperçu de la syntaxe - exemple

```
public class Complexe
{
    private double _reel, _imaginaire;

    public double Reel
    {
        get { return _reel; }
        set { _reel = value; }
    }

    public double Imaginaire
    {
        get { return _imaginaire; }
        set { _imaginaire = value; }
    }

    public void Afficher()
    {
        Console.Write("{0} + {1} . i", _reel, _imaginaire);
    }
```

Le langage C#

Aperçu de la syntaxe - exemple

```
    public static Complexe Somme(Complexe a, Complexe b)
    {
        Complexe resultat = new Complexe();
        resultat._reel = a._reel + b._reel;
        resultat._imaginaire = a._imaginaire + b._imaginaire;
        return resultat;
    }

    public static Complexe operator+(Complexe a, Complexe b)
    {
        Complexe resultat = new Complexe();
        resultat._reel = a._reel + b._reel;
        resultat._imaginaire = a._imaginaire + b._imaginaire;
        return resultat;
    }

}
```


Le langage C#

Aperçu - création de collections d'objets

- Tableaux

```
const int NombreMesures = 1000;
double mesures[] = new double[NombreMesures];
```

- Liste générique

```
List<double> liste;
liste = new List<double>();
liste.Add(1.5);
MessageBox.Show(liste[0].ToString());
```

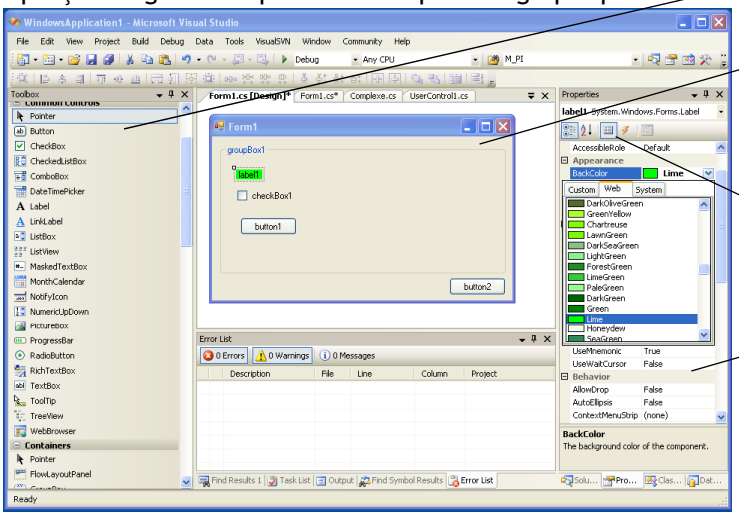
heig-vd

Le développement visuel WinForms
Aperçu

- Création interactive d'interfaces graphiques.
- Développement basé sur des objets
  - Composants graphiques
- Code généré automatiquement par l'outil de développement.

heig-vd

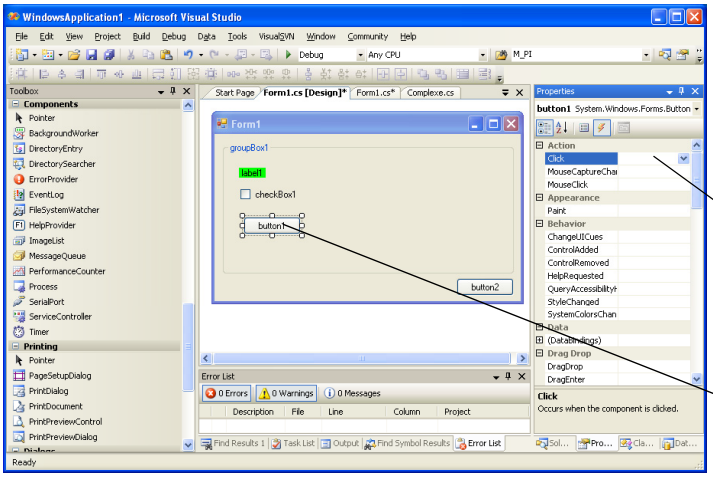
Le développement visuel WinForms
Aperçu – le glisser-déposer de composants graphiques



heig-vd

Le développement visuel WinForms

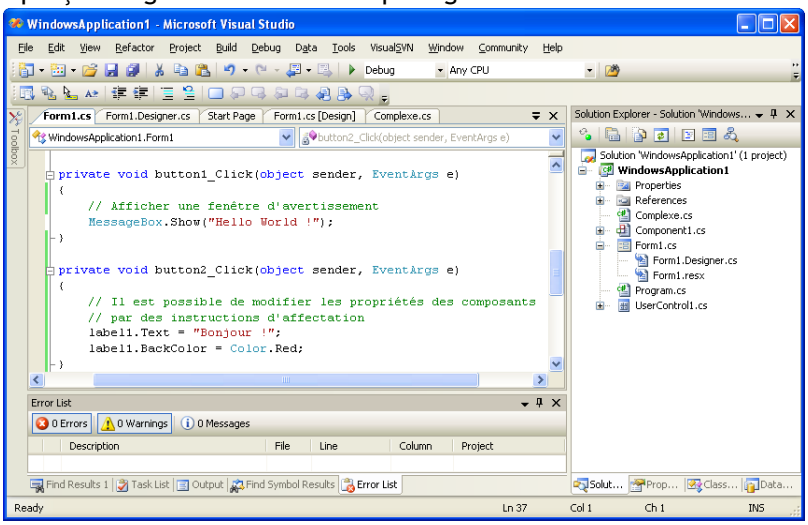
Aperçu – la cération de gestionnaires d'évènements



heig-vd

Le développement visuel WinForms

Aperçu – la génération de code pour gérer les évènements



Le développement visuel WinForms

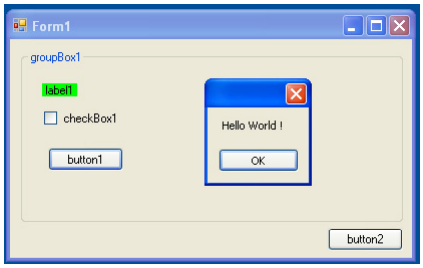
Aperçu – le code généré automatiquement pour les composants

```
#region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.BackColor = System.Drawing.Color.Lime;
            this.label1.Location = new System.Drawing.Point(21, 33);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(35, 13);
            this.label1.TabIndex = 0;
            this.label1.Text = "label1";
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(27, 97);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(75, 23);
            this.button1.TabIndex = 2;
            this.button1.Text = "button1";
            this.button1.UseVisualStyleBackColor = true;
```


Le développement visuel WinForms

Résultat – une application graphique Windows




Exercices
C# et .Net