Module .NET

Chapitre 2

Le .NET Framework

Christian ALLEMAND – 2011/2012

Contenu

Cours

2.1 - Définition de l'ECMA, ses acteurs, son rôle

Développé par Microsoft, le .NET Framework est né, d'une certaine façon, grâce aux travaux de l'ECMA (basé à Genève).

L'ECMA (European Computer Manufaturer's Association) a été créée en 1961 et regroupe les acteurs majeurs de l'informatique.
Elle a la charge d'établir des standards pour entre autres, les systèmes de communications et d'information : en bref elle établit des standards publics.
Standards publics car diffusés et consultables par tous.
Site de l'ECMA: http://www.ecma-international.org/

Elle est composée des catégories de membres suivantes :
Membres ordinaires avec droits de vote :
Adobe, AMD, Google, HP, IBM, Intel, Microsoft, Nvidia, Sony, …
Membres associatifs sans droit de vote
Appel, Boeing, Novell, Pioneer, Samsung, Siemens, …
Petites et moyennes entreprises (CA < 100 millions de francs suisses)
Petites entreprises (CA < 5 millions de francs suisses)
Association à but non lucratif (fondation Mozilla entre autres)

Le **domaine d'action** des différents **groupes de travail** de l'ECMA comprend entre autres :

- Les communications sans fils haute vitesse
- Stockage optique
- Format Office Open XML
- ECMAScript, comme le JScript (TC39)
- Les langages de programmation (TC49)
- …

Parmi tous les différents **comités techniques** qui composent cette association, l'un d'eux nous intéresse plus particulièrement : le TC49 qui s'occupe du theme **Programming ans Scripting Languages**.
Ses objectifs sont entre autres :
Développer un **standard** pour le **langage de programmation C#** (TG2) . Standard ECMA-334
Développer un **standard d'infrastructure de langage commun – CLI** (TG3) . Standard ECMA-335
Développer un standard pour le langage de programmation Eiffel (TG4) . Standard ECMA-367
Développer un standard de liaison entre C++ et la CLI (TG5) . Standard ECMA-372

Le standard qui va nous intéresser est le Standard **ECMA-335**

2.2 - La CLI et ses implémentations

2.2.1 - La Common Language Infrastructure (CLI)

L'objectif de la **CLI** (***Common Language Infrastructure*** ou Infrastructure de langage commun) est de créer une infrastructure dans laquelle des applications écrites dans différents langages de haut niveau peuvent être exécutées dans différents environnements systèmes sans avoir besoin d'adaptation au dit système.
Le Microsoft **.NET Framework** est une **implémentation de la CLI** sur les systèmes d'exploitation **Windows**.
Concrètement, une application ciblant le framework 2.0, compilée en **Intermediate Language** sur une plate-forme Windows pourra être exécutée sans modification par le runtime correspondant sur un système MAC ou LINUX sur lequel est installée une implémentation du framework correspondant.

Une application exécutable standard contient du code machine ciblant un système d'exploitation et un processeur en particulier.

Par exemple, si l'on souhaite commercialiser une application exécutable standard afin qu'elle puisse être exécutée sur 2 processeur différents (x86 et x64) et sur 2 OS (Windows et Linux), il faut réaliser 4 compilations différentes.

Avec l'Intermediate Language, il seule compilation sera nécessaire pour ces 4 combinaisons différentes car c'est le VES (le runtime CLI) installé sur l'ordinateur client qui se chargera de traduire l'Intermediate Language en code machine adapté au processeur et au système d'exploitation.

2.2.2 - Les implémentations de .NET Framework sur les OS autres que Windows

Aujourd'hui, cette implémentation du .NET Framework s'est étendue aux systèmes d'exploitation Linux et MAC par l'intermédiaire de différents projets.
On trouve entre autres :

DotGNU Project

DotGNU Portable.NET est une implémentation de la CLI pour les OS suivants
GNU/Linux (sur PCs, Sparc, iPAQ, Sharp Zaurus, PlayStation 2, Xbox,...), BSD, Cygwin/Mingw32, Mac OS X, Solaris, AIX
Source : http://www.gnu.org/s/dotgnu/

Mono (Novell) est une implémentation du .NET Framework sur Linux. Toutes les fonctionnalités du .NET Framework ne sont pas incluses.

MonoDevelop

MonoDevelop est un IDE conçu principalement pour C# et VB.NET et autres langages. NET.
MonoDevelop permet aux développeurs d'écrire rapidement des applications **Web ASP.NET** et de **bureau** sous **Linux**, Windows et Mac OSX.
Avec MonoDevelop, il est facile pour les développeurs de porter des applications. NET créées avec Visual Studio vers Linux ou Mac OSX et de maintenir une base de code unique pour toutes les plateformes.

Mono permet de déployer des applications ASP.NET sous Apache sur Linux.
Liens utile : http://vincentlaine.developpez.com/tuto/dotnet/mod_mono/#LIV-A

C'est un projet Open Source, donc gratuit, et dont on peut récupérer les sources.
Cette application s'installe :

- Sous Windows
- Sous Linux (openSUSE, SLE, Debian, Ubuntu)

- Sous Mac OSX

Actuellement en version stable 2.6 (avec support du .NET 4.0)
Nécessite Mono 2.10.4 et GTK# 2.12.8
Source : http://www.mono-project.com/Main_Page et http://monodevelop.com/

MonoTouch for iOS

Mono for Android permet de développer des applis pour iPhone et iPad en utilisant le C# et le .NET Framework (solution commercialisée aux environ de 400$).
Cette application s'installe :

- Sous Visual Studio (windows). Attention ne fonctionne pas avec Visual Studio Express
- Sous MonoDevelop (windows)
- Sous MonoDevelop (Mac OSX)

Source : http://ios.xamarin.com/

Mono for Android

Mono for Android permet de développer des applis pour Android en utilisant le C# et le .NET Framework (solution commercialisée aux environ de 400$).
Cette application s'installe :

- sous Visual Studio (windows). Attention ne fonctionne pas avec Visual Studio Express
- Sous MonoDevelop (windows)
- Sous MonoDevelop (Mac OSX)

Source : http://android.xamarin.com/

2.2.3 - Intégration du Framework dans les versions de Windows

A partir de 2002, Microsoft à commencer à intégrer le .NET Framework dans les versions les plus utilisées de son système d'exploitation Windows.

Version .NET	Numéro de version	Date de sortie	Visual Studio	Par défaut dans Windows
1.0	1.0.3705.0	13 février 2002	**Visual Studio .NET 2002**	Windows XP versions Tablette et Media Center
1.1	1.1.4322.573	24 avril 2003	**Visual Studio .NET 2003**	Windows Server 2003
2.0	2.0.50727.42	7 novembre 2005	**Visual Studio 2005**	Windows Server 2003 R2
3.0	3.0.4506.30	6 novembre 2006		Windows Vista, Windows Server 2008
3.5	3.5.21022.8	19 novembre 2007	**Visual Studio 2008**	Windows 7, Windows Server 2008 R2
4.0	4.0.30319.1	12 avril 2010	**Visual Studio 2010**	Windows Server 2008 R2 SP1

4.5	4.5	13 septembre 2011 (Developer Preview)	**Visual Studio '11'** **(Nom de code)**	Windows 8, Windows Server 8

Lorsque l'on installe explicitement une version du Framework, cette opération peut être assez longue.

La raison est que l'installation ne se limite pas à copier les outils et les librairies sur le disque.

En effet, lors de l'installation, toutes les librairies présentes dans le GAC (Global Assembly Cache) sont compilées une par une afin d'être optimisées en fonction de l'architecture de l'ordinateur sur lequel elles sont installées.

2.3 - Les éléments de la CLI

La CLI définit les éléments suivants : le CTS, les métadonnées, la CLS et le VES.

2.3.1 - Le Common Type System (CTS)

Le CTS (le système de type commun) fournit un système de type riche qui supporte de nombreux types et opérations supportés par de nombreux langages.
Il définit 2 entités principales:

- Les types par valeur (Value Type)
- Les types par référence (Object Type)

Il définit également le comportement de ces types, les règles du boxing et de l'unboxing, du casting, etc …
Il faut garder à l'esprit que tout ce que l'on fait avec VB.NET ou C# est de la manipulation de types.

Le CTS du .NET framework prend en charge les 5 catégories de types suivantes :

- Les Structures (type valeur)
- Les Enumérations (type valeur)
- Les Classes (type référence)
- Les Interfaces (type référence)
- Les Délégués (type référence)

Toutes les règles qui s'appliquent au CTS s'appliquent à la spécification CLS, sauf dans les cas où des règles plus strictes sont définies dans la spécification CLS.

Vous trouverez plus de détails à partir du chapitre sur le CTS du standard ECMA-335.

2.3.2 - Métadonnées

Les métadonnées sont utilisées par la CLI pour décrire les types et sont stockées de façon indépendante du langage.
Cette partie de la CLI définit comment ces métadonnées doivent être formatées, stockées et les règles pour y accéder.
Pour permettre au CLR (runtime) de fournir des services de code managé, les compilateurs doivent fournir des métadonnées qui décrivent les types, les membres, et des références du code.
Les métadonnées sont stockées avec le code.
Tout programme Portable Exécutable (PE) contient des métadonnées.
Le CLR utilise les métadonnées afin de :

- localiser et de charger les classes,
- organiser les instances en mémoire,
- résoudre les invocations de méthodes,
- générer le code natif,
- renforcer la sécurité et fixer les limites du contexte d'exécution.

On verra plus loin que grâce à cette notion, notre code sera sûr et rendra l'utilisation des dll très souple.

2.3.3 - Le Common Language Specification (CLS)

Le CLS définit les règles que tout langage de programmation pour la CLI doit respecter.
C'est le **minimum vital** pour qu'un objet développé dans un langage **puisse communiquer** avec un autre objet développé dans le même langage ou dans un autre langage.
Le CLS s'appuie sur les règles définies dans le CTS.

Une application qui **respecte les règles de la CLS** est appelé **conforme CLS** (ou **CLS Compliant**).

Par exemple, **les entiers 64 bits non signés (Uint64) ne sont pas conformes CLS**.
Ils sont quand même implémentés dans le .NET Framework, mais leur utilisation, selon le framework cible, pourra générer un message d'avertissement pendant la compilation pour signaler leur **non-conformité CLS**.
Visual Studio permet de désactiver ces messages d'avertissement en plaçant un attribut (ClsCompliant) devant la déclaration d'un type qui utilise ce type d'entier.
Pour une application autonome, ce n'est pas trop grave.
En revanche, si on développe une librairie qui peut être utilisée par la suite par une application fonctionnant sur autre système, le résultat n'est pas prévisible.

La spécification CLS a été conçue pour être **suffisamment grande** pour inclure les **constructions de langage** qui sont **fréquemment utilisées** par les développeurs, tout en étant **suffisamment petite** pour que la plupart des langages puissent la prendre en charge.

De plus, **toutes les constructions** de langage qui **empêchaient de vérifier** rapidement **la sécurité de type** du code ont été **exclues** de la spécification CLS.

C'est donc en partie **grâce à la CLS** que différents langages de programmation **peuvent « communiquer** » entre eux de **façon sure**.

2.3.4 - Le Virtual Engine System (VES /CLR)

C'est lui qui a la charge d'exécuter les applications compilées en code IL.
Le **CLR** (Common Language Runtime) est une implémentation du **VES** dans le .NET Framework de Microsoft.

Le CLR intègre entre autres :

- Le **CAS** (**Code Access Security**) qui contrôle **l'intégrité du code** (validation du code et des métadonnées) et qui vérifie que le code ne fait rien de dangereux.

- Le **GC** (Garbage Collector) qui est chargé de la gestion de la mémoire (allocation et libération).

- La **BCL** (Base Class Library) qui représente la librairie standard pour la CLI.

Elle comprend les types de base de la CLI telles que les types intégrés(entiers, doubles,…), les entrées/sorties, les attributs, les chaines de caractères et leur manipulation,le formatage, les fluxs, mes collections …
Ne pas confondre la BCL et la FCL (Framework Class Library).

- La **FCL** qui est est une énorme librairie qui contient tous les types qui ne font pas partie de la BCL.

Elle inclut par exemple :
ADO.NET (ActiveX Data Object pour .NET), qui est un ensemble de classes qui permettent d'accéder à des données relationnelles (SQL par exemple), XML et d'application. ADO.NET est utilisable aussi bien par des applications desktop que par des applications Web.

Windows Forms, qui permet de créer des interfaces utilisateurs à base de formulaires (fenetres) et de controles (textboxs, listes déroulantes, grilles de données, …). Windows Forms n'est utilisable pour des applications Desktop et pas pour des applications Web.

ASP.NET (Active Server Page pour .NET) successeur de la technologie Active Server Pages (ASP). ASP.NET est un ensemble de technologies de programmation web permettant de créer des sites web dynamiques, des applications web ou des web services XML. Afin de pouvoir utiliser ASP.NET, on utilise un serveur web compatible ASP (IIS) ou le serveur web de développement intégré dans Visual Studio.

WPF (Windows Presentation Foundation) (depuis .NET 3.0), qui est une surcouche permettant de réaliser les interfaces graphiques utilisateur à la place de Windows Forms. Une des différences les plus importantes avec ce dernier est que la description des graphismes, basée sur le langage XAML, est entièrement vectorielle. Principalement utilisé pour les applications desktop, il sert aussi de base aux applications Web Silverlight.

WCF (Windows Communication Foundation) (depuis .NET 3.0),
Qui permet de construire des applications distribuées, et de faire communiquer des composants applicatifs se trouvant sur une même machine ou différentes machines reliées en réseau.
WCF permet de faire communiquer des applications créées avec différentes technologies (Remoting, Web Services, COM+,...) sans devoir reprendre le développement de ces applications.

WF (Windows Workflow Foundation) qui est une infrastructure qui permet aux utilisateurs de créer des workflows système ou utilisateurs dans leurs applications. Il comprend un espace de noms, un moteur de workflow in-process et des concepteurs pour Visual Studio.

2.3.5 - Structure du .NET Framework