Programmation en C# sous Linux avec Mono

par Lainé Vincent

Date de publication : 09/05/2005

Dernière mise à jour :

Ce tutoriel a pour but de faire découvrir au lecteur le framework Mono qui permet de faire du développement en C# et VB.NET sous Linux

I - Introduction

.Net est un environnement managé utilisant un compilateur Just In Time (JIT) pour compiler à la volé le code IL (Intermediate Language) des assemblies formant les programmes écrits en .Net .

Le Framework gère en particulier l'allocation de mémoire et la sécurité des assemblies. De plus, il met en place le Global Assembly Cache (GAC) qui permet la centralisation et le contrôle des versions des librairies grâce à une signature cryptographique.

Tous ces éléments et en particulier le code managé fait de .Net un environnement potentiellement multiplates-formes. Toutefois, Microsoft (le créateur de .NET) n'a pas souhaité sortir le Framework .Net pour d'autres plates-formes comme Mac ou Unix/Linux.

C'est de cet oubli que viennent les projets Mono et DotGNU qui sont nés grâce à la disponibilité des spécifications. Dans ce tutoriel je ne parlerais pas de DotGNU tout simplement car je ne l'ai pas testé.

Au cours de ce premier tutoriel consacré à la programmation en C# avec Mono nous allons voir les bases de Mono.

Ce tutoriel suppose que vous avez les bases en programmation avec .Net et C#.


Attention afin de faciliter la compréhension je parlerais de Framework .Net pour le Framework (sic ...) distribué par Microsoft et de Mono pour le Framework Mono.

Mono supporte actuellement le C# en standard et de plus un compilateur VB.NET est en cours de programmation. La version présente dans le paquetage 1.1.4 est la version 0.92 de ce compilateur nommé mbas (Mono Basic)

II - Installation

Autant le dire tout de suite : l'installation de logiciel sous Linux n'est pas chose aisée.

Mono ne déroge pas à la règle pour peu que l'on veuille l'installer à partir des sources avec GTK#. Personnellement, je n'ai pas réussi à installer Mono 1.0.6 et GTK# à partir des sources sur ma Mandrake 10.1 (problème de dépendances).

J'ai donc opté pour les paquets disponibles sur les sources contrib des miroirs Mandrake. Toutefois il existe de nombreux paquets précompilés, pour de nombreuses distributions, disponibles sur le site de Mono et les heureux utilisateurs de Gentoo (comme moi :-)) trouveront les paquets portage pour leur distribution favorite.

De plus au moment de la rédaction de ce tutoriel des installateurs binaires de mono sont disponibles pour Windows et Linux. Sous Linux l'installateur ne requière aucune dépendance à l'installation mais l'exécution de certains programmes (comme MonoDoc) demande à ce que certains autres logiciels écrit dans d'autre langages comme C/C++ (GTK+, GECKO, ...) soient installés. Personnellement je trouve que ce paquet binaire est une bonne chose, car il facilite le déploiement de Mono sur les machines cibles. Il faut cependant faire attention aux dépendances cachées de vos programmes et donc les tester sur une installation minimale de Mono et d'une distribution Linux .

Par exemple, pour une application écrite avec GTK#, tester votre application sur machine qui ne possède pas GTK+ et Gnome afin de voir s'il y a des dépendances à combler.

Pour ceux qui sont sous Mac ou qui veulent installer Mono à partir des sources je les invite à ce rendre sur le site de Mono afin d'y trouver les informations nécessaires à la compilation et à l'installation.

Voilà un rapide récapitulatif des commandes :

```
$> tar xvfz mono-1.1.6.tar.gz
$> cd ./mono-1.1.6
$> ./configure
$> make
$> su
#> make install
#> exit
```

$> désigne le prompt du shell utilisateur et #> designe le prompt du shell root (ou super utilisateur)

Après cela Mono est installé.

Pour installer GTK# :

```
$> tar xvfz gtk-sharp-1.9.2.tar.gz
$> cd gtk-sharp-1.9.2
$> ./configure
$> make
$> su
#> make install
```

```
#> exit
```

En cas d'erreur de la configuration de GTK# vérifiez que les fichiers de configuration d'application (.pc) de mono sont bien installé au bon endroit.

Dans mon cas il a fallu que je les copie dans /usr/lib/pkgconfig/ car l'installation de mono les installés dans /usr/local/lib/pkgconfig

Après l'exécution de ces commandes, Mono est installé et configuré pour utiliser les winforms managées (voir plus loin) et GTK#

III - Ou se trouve quoi ?

Quels sont les éléments les plus importants que mono embarque ?

- mono : c'est l'exécutable le plus important, car il permet d'exécuter les applications .NET

- gacutil : c'est l'utilitaire permettant d'ajouter des assemblies au Global Assembly Cache

IV - Un petit tour de Mono

IV-A - Compatibilité avec le framework .NET

En théorie, la compatibilité entre le Framework .Net et Mono devrait être totale.

En effet, tous deux exploitent le code IL et les assemblies .Net sont compilés en IL.

Oui, mais voilà, les assemblies de base du Framework .Net font appel aux fonctions de l'API Win32, or ces fonctions ne sont pas présentent sous Linux ou Mac. C'est pourquoi Mono redéfinit les assemblies de base pour assurer la compatibilité avec les systèmes Linux et Mac.

Toutefois, les assemblies compilées sous Windows avec le Framework .Net, si elles ne comportent pas de référence(s) à des librairies que Mono n'a pas, peuvent tourner sans être recompilé sous Mono. Mais je ne pense pas que ce soit une bonne idée à moins que la compilation de votre librairie/programme prenne plus de deux jours.

En effet, les deux frameworks même s'ils exploitent les même langages, restent différents par certains points.

IV-B - Le cas des Winforms

Comme dit précédemment les classes de bases du Framework .NET ont été portées pour tourner sous Linux et Mac.

Toutefois il existe une classe importante qui n'a pu être porté, la classes System.Windows.Forms.

En effet cette classe est dédiée à la gestion de l'interface graphique et des signaux aux applications. Or le traitement des signaux et les composants standard de l'interface graphique sont forts différents sur les différentes plates formes. Ces nombreux appels aux fonctions de l'API WIN32 font que cette classe n'a pas pu être porté pour l'instant.

Mais le Projet MWF (Managed Windows Form) est un projet dynamique et dès maintenant la version 1.1.6 embarque les Winforms avec elle. Ce projet avance rapidement maintenant et à l'heure de l'écriture de ce tutoriel le projet est marqué comme fini à 93 %. C'est pourquoi je me permet de faire ici l'hypothèse que avant la fin de l'année 2005, les Winforms seront totalement supportées.

De plus j'ai réussi à faire tourner une petite application basique compilée sous windows avec le framework 1.1 sous mono avec les Winforms Managées .

IV-C - Mais s'il n'y a pas de Winforms avec quoi faire des interfaces graphiques ?

Pour palier au problème de la classe Winforms, l'équipe de Mono à décidé de créer une surcouche à GTK+ nommé GTK#.

Pourquoi avoir choisi GTK plutôt que Qt qui est écrit en C++ donc plus proche du C# ? Pour deux raisons : D'abord, Qt est un Framework commercial et surtout le chef de projet de Mono n'est autre que celui de Gnome (l'environnement de bureau) Miguel de Icaza.

Un projet pour créer une surcouche à Qt est en cours de développement, mais je n'ai pas pu le tester.

Plus généralement Mono est distribué avec de nombreux composants optionnels tels que les drivers pour MySql (déconseillé voir sur le site de MySql et l'exemple à la fin), Firebird (base de données), Mozilla firefox (GECKO), et surtout OpenGL ainsi que bien d'autre.

V - Un EDI pour Mono : MonoDevelop

Afin de développer des applications C# avec Mono sous Linux, un EDI a été développé.

Il s'appelle MonoDevelop et est présent dans les installateurs binaires de Mono pour Linux.

Pour développer avec Mono sous Windows, l'EDI le plus approprié est l'excellent SharpDevelop qui permet de choisir le compilateur et l'interpréteur.

V-A - Installation

Afin de procéder à l'installation de MonoDevelop, il vous faut récupérer soit les paquets binaires, soit les sources ou encore l'installateur binaire.

Ensuite je vous invite à vous reporter à l'explication qui est fourni sur le site de MonoDevelop afin de le compiler et de l'installer à partir des sources.

Si vous avez choisi l'une des deux autres méthodes, MonoDevelop est déjà installé.

V-B - ça ressemble à quelque chose de connu ...

En effet, MonoDevelop est le portage partiel de SharpDevelop sous GTK#/Mono. Je dis partiel car de nombreuses fonctionnalités de SharpDevelop ne sont pas présentes sous MonoDevelop.

V-C - Mais où est le concepteur graphique ?

Il a été simplement été supprimé, car il ne pouvait pas servir pour la création d'interfaces graphiques avec GTK#.

On ne peut qu'espérer qu'un outil de ce genre soit rapidement développé et intégré à MonoDevelop et apparement ce projet est en cours de réalisation, mais aucune précision n'a été trouvée

V-D - Les fichiers générés par MonoDevelop

À la création d'une nouvelle solution, MonoDevelop génère de nombreux fichiers. Toutefois, un seul va nous importer pour l'instant : c'est le fichier Makefile que tout 'Linuxien' connaît.

Je tenais à le préciser, car c'est un fichier important pour la compilation et surtout qui mérite d'être examiné afin de comprendre comment fonctionne le compilateur mcs.

Attention : Dans la version 0.6 de MonoDevelop, les fichiers cités au-dessus ne sont pas générés automatiquement.

De plus, le format des fichiers de solutions à été modifié pour des raisons internes.

VI - Programmes console avec Mono.

Nous allons voir maintenant une série d'applications toutes simples pour illustrer le développement avec Mono.

VI-A - Le fameux « Hello World »

Ce premier programme va simplement nous servir à illustrer la compilation en ligne de commandes et les fichiers Makefile présents dans les répertoires des programmes.

Je tiens à ajouter une petite précision au sujet des fichiers Makefile contenus dans les répertoires des exemples.

Tout a l'heure, j'ai parlé des fichiers générés par MonoDevelop, toutefois les fichiers présents dans les exemples sont entièrement écrits à la main.

Le Hello world

```
using System;

namespace Hello_World
{
 public class Hello_World
 {
        public Hello_World()
        {

        }

        public static void Main(string[] args)
        {
              Console.WriteLine("Hello Linux World :-) ");

        }

 }
}
```

Ligne de commande pour compiler avec mcs

```
mcs -out:hello_world.exe ./src/*.cs
```

Voyons maintenant les options utilisées et celle qui ne le sont pas encore dans les exemples :

Commandes	Description
-reference: ou -r:	assembly référencé dans le programme. L'assembly doit etre dans le GAC
-lib: ou -l:	La même chose mais l'assembly ne doit pas etre dans le GAC
-pkg:	lie le package (au sens Linux) avec le programme à compiler (Exemple : GTK#)
-target ou /t:	la cible de la compilation: exe, winexe,library,module
-out:	nom du fichier de sortie

Ce tableau n'est qu'un petit aperçu des options du compilateur mcs. Si vous voulez toutes les connaître vous faites :

```
mcs --help
```

```
ou
man mcs
```

Je vais maintenant vous montrer rapidement un fichier Makefile que j'utilise pour compiler sous Linux/Mono

```
all:
	mcs -out:MonoMySqlIO.exe -r:./MySql.Data.dll -r:System.Data.dll ./MonoMySqlIO/*.cs
	echo "mono ./MonoMySqlIO.exe" > ./monoMySqlIO
	chmod +x ./monoMySqlIO
```

La première déclare le nom de la règle pour les commandes.

La deuxième ligne effectue la compilation en elle-même.

mcs est le compilateur C# de Mono.

-out:MonoMySqlIO.exe indique que le nom du fichier de sortie du compilateur

-r:./MySql.Data.dll indique que l'on doit lier l'assembly MySql.Data.dll situé dans le répertoire courant (./) au programme

./MonoMySqlIO/*.cs indique l'emplacement des fichiers pour la compilation

La troisième ligne crée un fichier qui va permettre de lancer facilement le programme

echo "mono ./MonoMySqlIO.exe" Permet d'afficher sur la sortie standard le texte entre ""

> ./monoMySqlIO redirige la sortie standard vers le fichier monoMySqlIO

La quatrième ligne rend le fichier monoMySqlIO exécutable

chmod commande standard de Linux pour changer les droit sur un fichier

-x ajoute le droit d'exécution pour l'utilisateur

./monoMySqlIO le fichier sur lequel on applique ces nouveaux droits

VI-B - Accès à MySql avec Mono

Dans cet exemple, nous allons voir comment accès à une base MySql en C#. Le driver utilisé est celui de MySql.com (version compilé)

Accès à MySql sous Mono/Linux

```
using System;
using System.Data;
using MySql.Data.MySqlClient;
```

Accès à MySql sous Mono/Linux

```
namespace MonoMySql
{
	class MainClass
	{
		private MySqlConnection connection;
		private MySqlCommand command;
		private MySqlDataReader dataReader;

		private bool Open;

		public MainClass()
		{
			this.Open = false;

		}

		public void Lunch()
		{
			Console.WriteLine("Entrez le nom du server :");
			string ServerName = Console.ReadLine();
			Console.WriteLine("Entrez le nom d'utilisateur");
			string userName = Console.ReadLine();
			Console.WriteLine("Entre le mot de passe");
			string password = Console.ReadLine();
			Console.WriteLine("Entrez la base ? laquel ce connecter ");
			string dataBase = Console.ReadLine();

			//On ouvre la connection avec les parametres rentre par l'utilisateur
			this.connection = new MySqlConnection("Server=" + ServerName + ";Database="
			+ dataBase + ";User ID="+ userName + ";Password=" + password + ";");

			try
			{
				this.connection.Open();
				this.Open = true;

				this.Menu();

			}
			//On verifie que MySql ne leve pas d'exception
			catch(MySqlException sqlEx)
			{
				Console.WriteLine("Exception MySql :-(");
				Console.WriteLine(sqlEx.Message);
				return ;
			}
			catch(Exception ex)
			{
				Console.WriteLine("Exception generale :-( ");
				Console.WriteLine(ex.Message);
				return;
			}



		}

		private void Menu()
		{
			bool exit = false;

			while(exit == false)
			{
				Console.WriteLine("Menu :");
				Console.WriteLine("----------------------");
				Console.WriteLine("1 : Executer une requete SELECT)");
				Console.WriteLine("2 : Executer une requete UPDATE,INSERT,CREATE,...");
```

Accès à MySql sous Mono/Linux

```
                Console.WriteLine("3 : Quitter");
                Console.WriteLine("Choix ?");
                int rep = Int32.Parse(Console.ReadLine());

                switch(rep)
                {
                    case 1 : this.ExecuteQuery();
                        break;
                    case 2 : this.ExecuteNoQuery();
                        break;
                    case 3 : exit = true;
                        break;
                    default : break;
                }

            }
        }

        private void ExecuteQuery()
        {
            Console.WriteLine("Entrez la requete a effectuer");
            string req = Console.ReadLine();

            //On cree une nouvelle command avec la requete a effecteur et on l'associe a la connection ouverte
            this.command = new MySqlCommand(req,this.connection);

            try
            {
                this.dataReader = this.command.ExecuteReader();

                while(this.dataReader.Read())
                {
                    foreach(Object obj in this.dataReader)
                    {
                        Console.Write(obj.ToString() + " *** ");
                    }
                    Console.WriteLine("----------------------------------------");
                }
            }
            catch(MySqlException sqlEx)
            {
                Console.WriteLine(sqlEx.Message);
            }
            catch(Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
            finally
            {
                this.dataReader.Close();
            }


        }

        private void ExecuteNoQuery()
        {
            Console.WriteLine("Entrez votre requete");
            string req = Console.ReadLine();

            //On cree une nouvelle commande avec la requete et la connection ouverte
            this.command = new MySqlCommand(req,this.connection);

            try
            { //ExecuteNonQuery renvois le nombre de ligne affecte par la requete
                Console.WriteLine("Nombre de ligne affectee : " + this.command.ExecuteNonQuery());
            }
            catch(MySqlException sqlEx)
```

Accès à MySql sous Mono/Linux

```
                {
                        Console.WriteLine(sqlEx.Message);
                }
                catch(Exception ex)
                {
                        Console.WriteLine(ex.Message);
                }


        }

        public void CloseMySql()
        {
                if(this.Open == true)
                {
                        this.connection.Close();
                }
        }

        [STAThread]
        static void Main(string[] args)
        {
                Console.WriteLine("Bienvenue dans MonoMySql");
                MainClass mainClass = new MainClass();
                mainClass.Lunch();

                mainClass.CloseMySql();
                Console.WriteLine("A bientot dans MonoMySql");
        }
    }
}
```

VI-C - Les IO avec Mono, amélioration du programme d'accès à MySql

Dans cet exemple, nous allons voir la création et l'écriture dans un fichier

Changement par rapport à la version précédante

```
private FileStream fileStream;
private StreamWriter sw;

...

private void RunLog()
{
        this.fileStream = new FileStream("./sql.log",FileMode.Create);
        this.sw = new StreamWriter(this.fileStream);
        this.log = true;
}

private void SaveLog(string log)
{
        this.sw.WriteLine(log);
}

private void StopLog()
{
        if(this.log == true)
        {
                this.sw.Close();
                this.fileStream.Close();
                this.log = false;
        }
}
```

Conclusion sur ces programmes console avec Mono.

À l'issue de ces trois petits programmes, les développeurs qui connaissent un peu la programmation sous .NET me diront :

« Eh alors ou est la différence ? ».

Justement, ces programmes n'ont d'autre but que de montrer que si on sait programmer avec .NET sous Windows on sait le faire avec Mono sous Linux et Mac.

Il n'existe que très peu de différences entre les deux environnement de programmation et c'est un jeu d'enfant que de faire des programmes qui sont multiplates-formes pour peu que l'on sépare l'interface graphique du corps du programme.

Voici un exemple de portage d'une application sous Mono et GTK# et qui été écrite à l'origine sous .NET.

Une application portée sous Linux/Mac avec mono et GTK#

VII - Conclusion

Mono est un produit fini qui mérite que l'on passe un peu de temps à le découvrir.

En effet, il permet de porter des applications écrites en C# sous Linux et Mac sans trop de problèmes à condition, pour l'instant, de ne pas avoir d'interfaces graphiques trop complexes.

À noter qu'au moment de l'écriture de cet article une nouvelle version de Mono est sorti avec le support des Winforms en code managé (C#). Toutefois, par manque de temps je n'ai pas pu la tester complétement, mais cela prouve que le projet avance. Reste que si vos projets sont bien compartimentés, Mono est une solution viable pour faire tourner vos applications C# sous d'autres plates-formes que Windows.

À savoir également que Mono implémente d'ores et déjà les spécifications du Framework 2.0 de Microsoft alors même que celui-ci n'est pas encore sorti.

De plus même si dans ce tutoriel seul la partie « Windows » a été abordée Mono embarque dans son paquet un petit serveur ASP.NET et est tout à fait capable d'héberger des sites ASP.NET. Mais les utilisateurs d'ASP.NET seront d'avantage satisfais par « mod_mono » qui est un module d'Apache 2 pour supporter ASP.NET. Vous pouvez ainsi bénéficier de la stabilité et de la puissance d'Apache sous un serveur Linux en faisant tourner vos pages ASP.NET

En résumé :

Essayez Mono. Vous ne serez pas déçu par le logiciel en lui-même, mais plutôt par le manque d'offre de solution de développement qui fait que Mono tarde à prendre toute l'ampleur qu'il devrait.

Je souhaite longue vie à Mono qui m'a permis de concilier mes deux passions informatiques : Linux et C#.

VIII - Ressources

Voici les différentes addresses ou vous trouverez les logiciels :

Mono

MonoDevelop

GTK#

Liens autre en rapport avec les exemples :

Le driver .NET de MySql (utilisable sous Mono et Windows)

Liens interne pour la programmation

L'article de Thomas Lebrun sur MySql et .NET

Liens pour télécharger les projets (MonoDevelop)

le hello world

Le projet pour se connecter à MySql

Le projet MySql avec les fichiers de log

L'article au format pdf :

Mono.pdf

IX - Remerciements

Je tiens à remercier Cerberes pour la correction du français et tous ceux qui m'ont aidé à mettre au point cet article en particulier Laurent Dardenne pour ses nombreuses remarques.