Mono & Moonlight

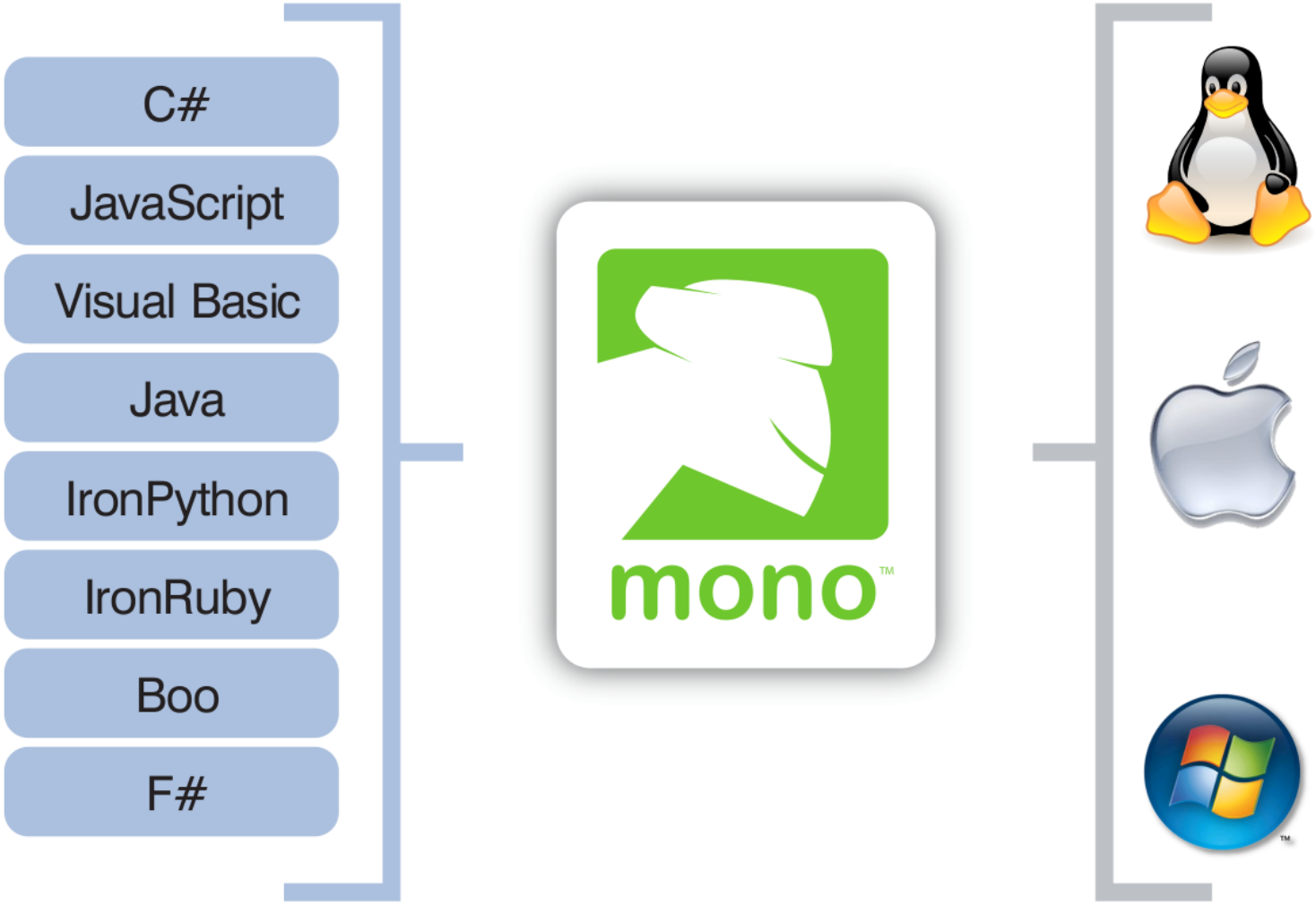
.net et Silverlight, différemment

Jb Evain
Software Engineer
jbevain@novell.com

Novell®

Mono

Multi plateformes, Multi langages

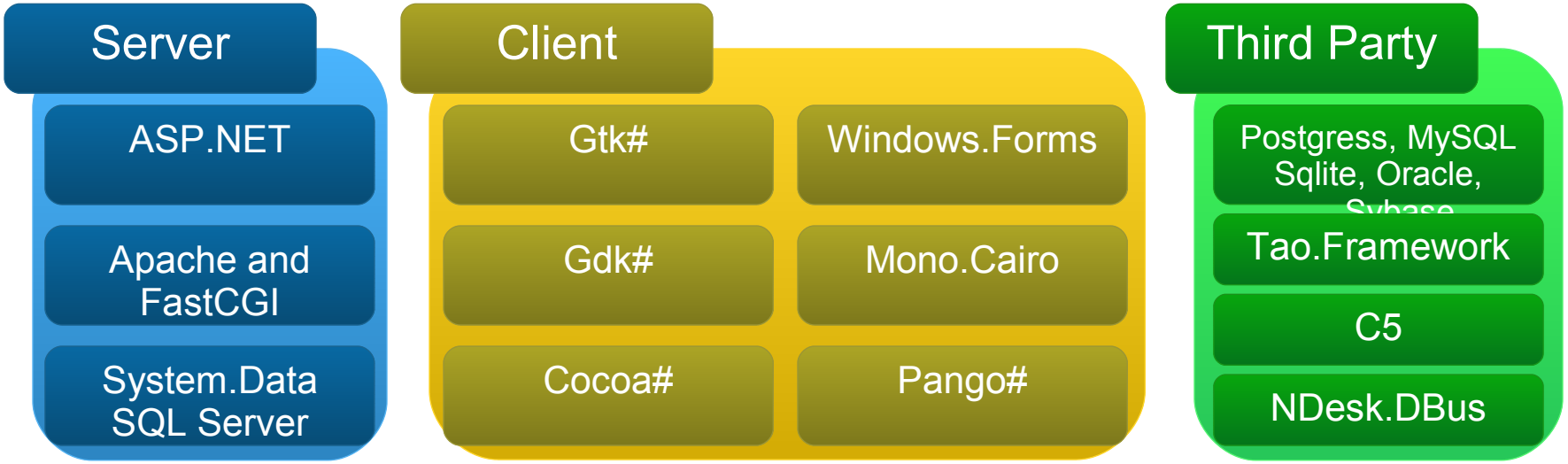


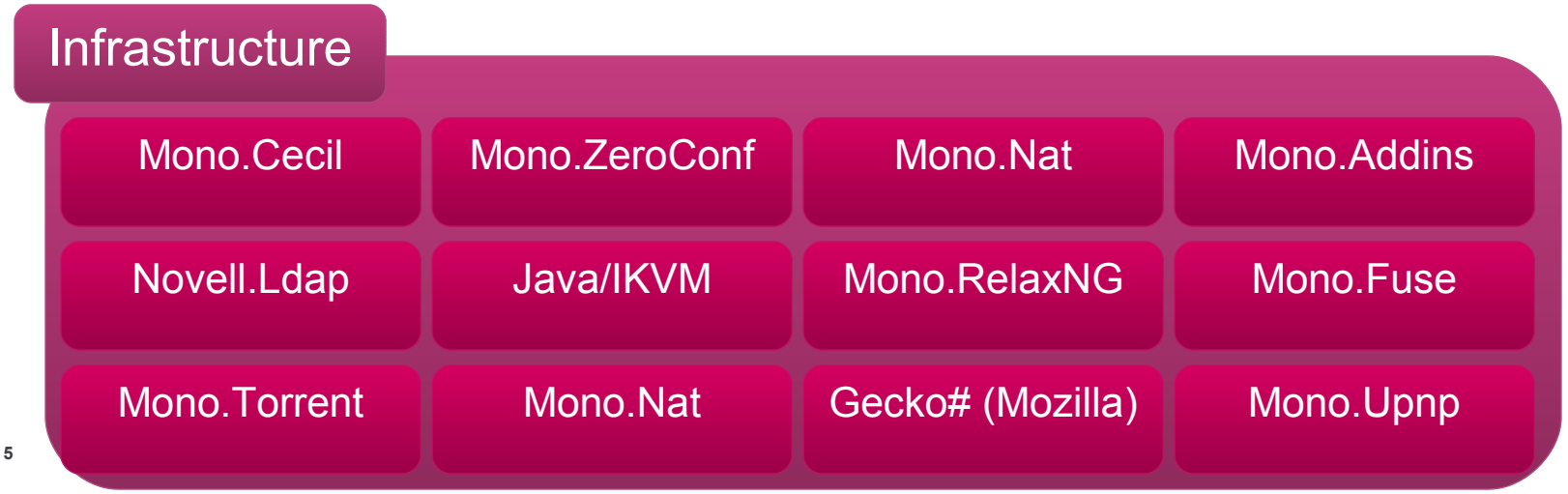
Multi plateformes, Multi langages

- C#
- JavaScript
- Visual Basic
- Java
- IronPython
- IronRuby
- Boo
- F#

mono™

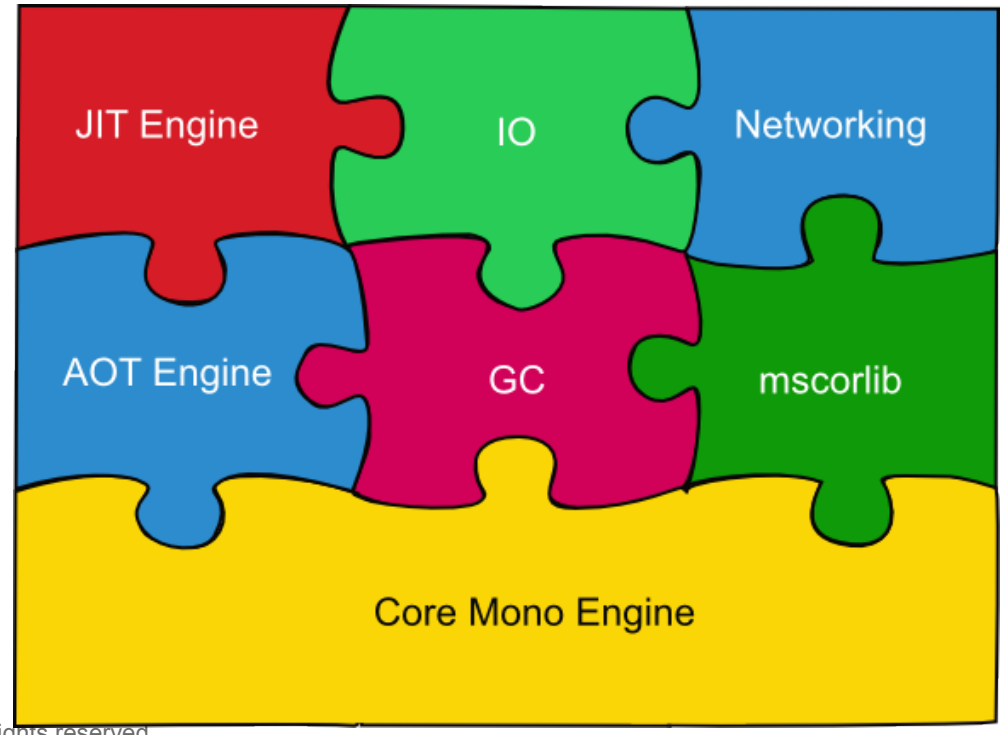
APIs


Server
ASP.NET
Apache and FastCGI
System.Data SQL Server
Client
Gtk#
Windows.Forms
Gdk#
Mono.Cairo
Cocoa#
Pango#
Third Party
Postgress, MySQL Sqlite, Oracle,
Tao.Framework
C5
NDesk.DBus


Infrastructure
Mono.Cecil
Mono.ZeroConf
Mono.Nat
Mono.Addins
Novell.Ldap
Java/IKVM
Mono.RelaxNG
Mono.Fuse
Mono.Torrent
Mono.Nat
Gecko# (Mozilla)
Mono.Upnp

Runtime Modulaire

- Complet: 100mo
- Minimal: 2mo



Développement

- 35 personnes chez Novell
- Des centaines de contributeurs

Mono 2.2

- .net 2.0
  - ADO.NET 2.0
  - ASP.NET 2.0
  - Windows.Forms 2.0
- .net 3.5
  - C# 3.0
  - LINQ to Objects
  - LINQ expression trees
  - LINQ to XML

JIT

- Linear IL: nouveau JIT
  - Nouvelle représentation intermédiaire du code
  - Simplifie l'écriture d'optimisations pour le JIT
- Effets
  - Taille du code natif généré entre 12 et 20% plus petit.
  - Génère du code plus optimisé, entre 10 et 30% plus rapide.
- Generic Sharing:
  - Réutilisation du code natif pour les types génériques compatibles (List<string> et List<object> par exemple).

Full AOT

- Ahead of Time compilation
  - L'équivalent de ngen
  - Pre-compile l'IL en code natif
- Effets
  - Reduit le temps de démarrage d'une application
  - Produit du code légèrement plus lent que le JIT.
- Full AOT
  - Utilisé sur certains appareils qui n'autorisent pas les JIT.

Mono.Simd

- Simple Instruction Multiple Data
- Librairie managée
- Reconnue par le JIT, qui va remplacer les appels managés par du code natif optimisé.
- Support de SSEx sur x86
- Support d'Altivec pour PPC prévu pour plus tard.

Mono.Simd

```
void UpdatePos (Vector3f [] points, Vector3f delta)
{
    for (int i = 0; i < points.Length; i++)
        points [i] += delta;
}
```

```
Vector3f static operator + (Vector3f a, Vector3f b)
{
    return new Vector3f (a.x+b.x, a.y+b.y, a.z+b.z);
}
```

Mono.Simd, sans

```
00000000 <X_UpdatePos>:
  0:  55              push   %ebp
  1:  8b ec           mov    %esp,%ebp
  3:  53              push   %ebx
  4:  57              push   %edi
  5:  56              push   %esi
  6:  83 ec 38        sub    $0x38,%esp
  9:  8b 75 08        mov    0x8(%ebp),%esi
  c:  8b 7d 0c        mov    0xc(%ebp),%edi
  f:  33 db           xor    %ebx,%ebx
 11:  e9 ad 00 00 00  jmp    c3 <X_UpdatePos+0xc3>
 16:  8b c0           mov    %eax,%eax
 18:  39 5e 0c        cmp    %ebx,0xc(%esi)
 1b:  0f 86 b5 00 00 00  jbe    d6 <X_UpdatePos+0xd6>
 21:  8b cb           mov    %ebx,%ecx
 23:  c1 e1 04        shl    $0x4,%ecx
 26:  8b c6           mov    %esi,%eax
 28:  03 c1           add    %ecx,%eax
 2a:  05 10 00 00 00  add    $0x10,%eax
 2f:  89 45 bc        mov    %eax,-0x44(%ebp)
 32:  8b 08           mov    (%eax),%ecx
 34:  89 4d c4        mov    %ecx,-0x3c(%ebp)
 37:  8b 48 04        mov    0x4(%eax),%ecx
 3a:  89 4d c8        mov    %ecx,-0x38(%ebp)
 3d:  8b 48 08        mov    0x8(%eax),%ecx
 40:  89 4d cc        mov    %ecx,-0x34(%ebp)
 43:  8b 40 0c        mov    0xc(%eax),%eax
 46:  89 45 d0        mov    %eax,-0x30(%ebp)
 49:  8b 07           mov    (%edi),%eax
 4b:  89 45 d4        mov    %eax,-0x2c(%ebp)
 4e:  8b 47 04        mov    0x4(%edi),%eax
 51:  89 45 d8        mov    %eax,-0x28(%ebp)
 54:  8b 47 08        mov    0x8(%edi),%eax
 57:  89 45 dc        mov    %eax,-0x24(%ebp)
 5a:  8b 47 0c        mov    0xc(%edi),%eax
 5d:  89 45 e0        mov    %eax,-0x20(%ebp)
 60:  8d 45 e4        lea    -0x1c(%ebp),%eax
 63:  83 ec 10        sub    $0x10,%esp
 66:  8b 4d d4        mov    -0x2c(%ebp),%ecx
 69:  89 0c 24        mov    %ecx,(%esp)
 6c:  8b 4d d8        mov    -0x28(%ebp),%ecx
 6f:  89 4c 24 04     mov    %ecx,0x4(%esp)
 73:  8b 4d dc        mov    -0x24(%ebp),%ecx
 76:  89 4c 24 08     mov    %ecx,0x8(%esp)
 7a:  8b 4d e0        mov    -0x20(%ebp),%ecx
 7d:  89 4c 24 0c     mov    %ecx,0xc(%esp)
 81:  83 ec 10        sub    $0x10,%esp
 84:  8b 4d c4        mov    -0x3c(%ebp),%ecx
 87:  89 0c 24        mov    %ecx,(%esp)
 8a:  8b 4d c8        mov    -0x38(%ebp),%ecx
 8d:  89 4c 24 04     mov    %ecx,0x4(%esp)
 91:  8b 4d cc        mov    -0x34(%ebp),%ecx
 94:  89 4c 24 08     mov    %ecx,0x8(%esp)
 98:  8b 4d d0        mov    -0x30(%ebp),%ecx
 9b:  89 4c 24 0c     mov    %ecx,0xc(%esp)
 9f:  50              push   %eax
 a0:  e8 43 00 00 00  call   op_Addition
 a5:  83 c4 20        add    $0x20,%esp
 a8:  8b 45 bc        mov    -0x44(%ebp),%eax
 ab:  8b 4d e4        mov    -0x1c(%ebp),%ecx
 ae:  89 08           mov    %ecx,(%eax)
 b0:  8b 4d e8        mov    -0x18(%ebp),%ecx
 b3:  89 48 04        mov    %ecx,0x4(%eax)
 b6:  8b 4d ec        mov    -0x14(%ebp),%ecx
 b9:  89 48 08        mov    %ecx,0x8(%eax)
 bc:  8b 4d f0        mov    -0x10(%ebp),%ecx
 bf:  89 48 0c        mov    %ecx,0xc(%eax)
 c2:  43              inc    %ebx
 c3:  8b 46 0c        mov    0xc(%esi),%eax
 c6:  3b d8           cmp    %eax,%ebx
 c8:  0f 8c 4a ff ff ff  jl     18 <X_UpdatePos+0x18>
 ce:  8d 65 f4        lea    -0xc(%ebp),%esp
 d1:  5e              pop    %esi
 d2:  5f              pop    %edi
 d3:  5b              pop    %ebx
 d4:  c9              leave
 d5:  c3              ret
```

Mono.Simd, avec

```
00000000 <X_UpdatePos>:
  0:   55                   push   %ebp
  1:   8b ec                mov    %esp,%ebp
  3:   53                   push   %ebx
  4:   57                   push   %edi
  5:   56                   push   %esi
  6:   83 ec 04             sub    $0x4,%esp
  9:   8b 75 08             mov    0x8(%ebp),%esi
  c:   8b 7d 0c             mov    0xc(%ebp),%edi
  f:   33 db                xor    %ebx,%ebx
 11:   eb 29                jmp    3c <X_UpdatePos+0x3c>
 13:   8d 64 24 00          lea    0x0(%esp),%esp
 17:   90                   nop
 18:   39 5e 0c             cmp    %ebx,0xc(%esi)
 1b:   0f 86 2a 00 00 00    jbe    4b <X_UpdatePos+0x4b>
 21:   8b cb                mov    %ebx,%ecx
 23:   c1 e1 04             shl    $0x4,%ecx
 26:   8b c6                mov    %esi,%eax
 28:   03 c1                add    %ecx,%eax
 2a:   05 10 00 00 00       add    $0x10,%eax
 2f:   0f 10 00             movups (%eax),%xmm0
 32:   0f 10 0f             movups (%edi),%xmm1
 35:   0f 58 c1             addps  %xmm1,%xmm0
 38:   0f 11 00             movups %xmm0,(%eax)
 3b:   43                   inc    %ebx
 3c:   8b 46 0c             mov    0xc(%esi),%eax
 3f:   3b d8                cmp    %eax,%ebx
 41:   7c d5                jl     18 <X_UpdatePos+0x18>
 43:   8d 65 f4             lea    -0xc(%ebp),%esp
 46:   5e                   pop    %esi
 47:   5f                   pop    %edi
 48:   5b                   pop    %ebx
 49:   c9                   leave
 4a:   c3                   ret
```

Compiler Service

- Compilateur C# écrit en C#
- Mono.CSharp.Evaluator
  - Evaluation C# dynamique

Mono.CSharp

- REPL
- Scripting
- Prototypage

Demo - csharp

Mono.Management

- Mono.Attach.VirtualMachine
  - Connection à un processus Mono,
  - Dans l'AppDomain racine,
  - Dans un nouveau thread.
- Permet d'obtenir des informations sur l'état d'un programme.
- Automation de programmes qui n'ont pas été pensés pour à la base.

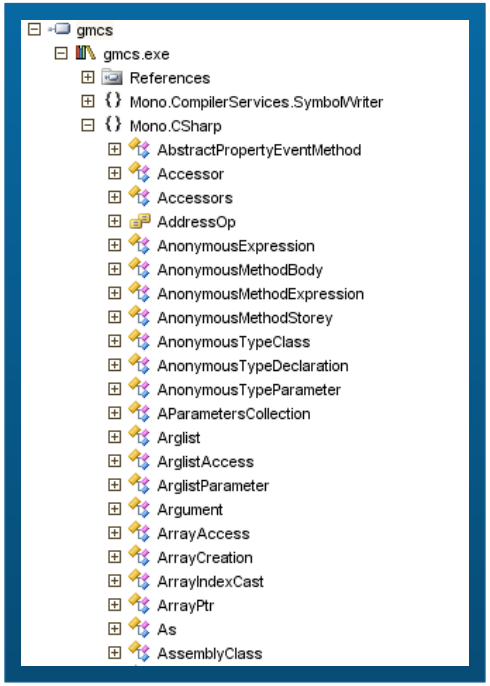
Demo - gsharp

Mono Linker, Mono Tuner

- Outils d'optimisation et de manipulation d'assemblies
- Le Linker enlève tout ce qui n'est pas nécessaire à une application
- Le Tuner applique des transformations

Mono Linker, Mono Tuner, example

Complete C# Compiler



Mono Linker

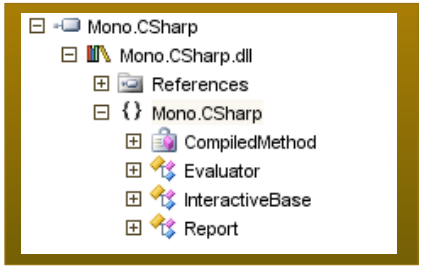
- Mono.Cecil
- Modifie le code
- Garde le nécessaire

Mono Tuner

- Ajuste l'api

link.xml

Mono.CSharp



MonoDevelop 2.0

- IDE complet
- Compatibilité Visual Studio sln, .csproj
- Code Completion
- Refactoring
- Addins
  - SGBDR
  - Boo, Java, Python, Nemerle, F#, …
- Debugger

mdb

- Mono a enfin un debugger.
- Interface en ligne de commande.
- Intégration MonoDevelop.
- Intégration Visual Studio en développement.

Demo - MonoDevelop

Gendarme

- Analyse statique à la FxCop
- Wizard Windows.
- 210 règles
  - BadPractice
  - Concurrency
  - Correctness
  - Design
  - Interoperability
  - Maintainability
  - Naming
  - Performances
  - Portability
  - Security

MoMA

- Outil d'aide à la portabilité
- Analyse statique des assemblies
- Potentiels problèmes:
  - P/Invoke,
  - Méthode non existante,
  - MonoTODO,
  - NotImplementedException

Moonlight

Moonlight 1.0

- Implémente Silverlight 1.0
- N'utilise pas Mono
- http://go-mono.com/moonlight/

Accord Novell, Microsoft

- Microsoft doit fournir:
  - Les spécifications
  - Les tests
  - Les codecs
- Novell doit fournir:
  - Une implémentation qui passe les tests
  - Et qui fonctionne sur les plus grosses distributions Linux

Moonlight 2.0

- Cible Silverlight 2.0
- Utilise Mono pour exécuter le code managé.
- beta pour MIX/09
- Intégration avec le desktop.

Conclusion

Liens

- http://www.mono-project.com
- http://www.mono-project.com/Start
- http://www.go-mono.com/monologue
- http://evain.net/blog/

Questions


Novell®