The .NET Platform and C#

Michael Hansen
Virtual Machine

- Common Language Runtime (CLR)
  - Windows 98 and up (Microsoft)
  - Mac OS X, Linux, BSD, Solaris, Wii, PS3, iPhone (Mono)
  - Browser plugin (Silverlight, Moonlight)
- Just-in-time compilation (JIT)
  - Code compiled on first use, optimized for local machine
  - Static compilation available (NGEN, Mono AOT)
- Common Language Infrastructure (CLI)
  - Security (sandboxing, trust levels)
  - Memory management (garbage collection)
  - Exception handling
Software Library

- **Base class library (BCL)**
  - Large collection of commonly needed libraries
  - I/O, GUI, networking, web development, numeric, cryptography, database, imaging, utility, interoperability
  - Subsets available in non-desktop .NET versions

- **3rd party libraries**
  - GTK# - Cross-platform GUI
  - IKVM.NET – JVM and core libraries for .NET
  - NPlot – Charting, targets in-memory bitmaps
  - OpenTK – OpenGL, OpenAL, 3-D maths
  - QuickGraph – Managed port of BGL
.NET Versions

The .NET Framework Stack

- Parallel LINQ
- Task Parallel Library
- LINQ
- ADO.NET Entity Framework
- WPF
- WCF
- WF
- Card Space
- WinForms
- ASP.NET
- ADO.NET
- Base Class Library
- Common Language Runtime

CLR Version 2.0

- Added Generics
- Adding DLR, STM (maybe)

Supported on Mono
Implementations

- Microsoft .NET (Windows only)
  - Gold standard
  - Desktop > Client Profile > Compact > Micro
- Mono (cross-platform)
  - .NET infrastructure + extras
  - Includes GTK#, SIMD library, SQLite, etc.
- Rotor (shared source CLI)
  - Microsoft “open source” implementation of CLI
  - Subset of .NET 2.0 BCL included
- DotGNU (not dead yet)
  - FSF implementation
Assemblies

- Basic unit of code
  - Includes both executables and libraries
  - Contains metadata, signature (optional)
- Common Intermediate Language (CIL)
  - Virtual machine code for CLR
  - All languages compile down to CIL
- Common Type System (CTS)
  - Rules for cross-language type safety
  - Object-oriented
  - Value types – structs, on stack
  - Reference types – pointers, on heap
Deployment (1 of 2)

- Stand-alone desktop (cross-platform)
  - Runtime must be present
  - Native libraries, additional assemblies bundled
- ClickOnce (Windows only)
  - Web-based deployment (like Java Web Start)
  - Automatic updates, integrates with client shell
  - Sandboxed, installed per-user
- Silverlight (cross-platform)
  - Browser plugin (Moonlight on Linux)
  - CLR, DLR, subset of BCL
  - WPF-based GUI, video/audio controls
Deployment (2 of 2)

- ASP.NET (cross-platform)
  - Web application framework
    - mod_mono on Apache
  - Web service hosting (WCF)

- Windows Mobile
  - Compact Framework needed

- MonoTouch
  - iPhone, iPod Touch, iPad (yes, already)

- Xbox 360
  - XNA Game Framework
  - Windows XP and up
C# Language (1 of 3)

- Interfaces
  - Poor man’s concepts
  - Explicit inheritance required

- Reflection
  - Access metadata for any object at runtime

- Generics
  - Poor man’s templates
  - Constraints
C# Language (2 of 3)

- Extension Methods
  - Static methods -> instance methods
    ```csharp
    static string Capitalize(this string s)
    {
      return (Char.ToUpper(s) + s.Substring(1));
    }
    
    "mike".Capitalize() // -> "Mike"
    ```

- LINQ, Anonymous Types
  ```csharp
  from emp in employees
  where (emp.Performance > 5) &&
  (emp.felonies.Count == 0)
  orderby emp.YearsOfService desc
  select new {
    Id = emp.Id,
    Name = emp.Last + ", " + emp.Last
  };
  ```
C# Language (3 of 3)

- **λ-expressions**

```csharp
1 Enumerable.Range(0, 10).Select(x => x * x);
```

- **Synchronization**
  - Any reference type

```csharp
1 lock (anyObject) {
2     // Access to anyObject is synchronized
3 }
```

- **Unsafe code (Full Trust required)**

```csharp
1 void BadCopy(byte[] src, byte[] dest) {
2     fixed (byte* pSrc = src, pDest = dest) {
3         for (int i = 0; i < src.Length; ++i) {
4             *pSrc = *pDest;
5         }
6     }
7 }
```
Threading (1 of 2)

- .NET Version <= 3.5
  - Explicit management
    - Create, destroy manually

```csharp
void ThreadProc() {
    // ...
}

new Thread(ThreadProc).Start();
```

- Thread pool
  - User work items, asynchronous callbacks

```csharp
void Callback(IAsyncResult result) {
    var clientSocket = (Socket)serverSocket.EndAccept(result);
}

serverSocket.BeginAccept(Callback, state);
```
Threading (2 of 2)

- **NET 4.0**
  - Parallel LINQ, for, foreach (IEnumerable)
    ```csharp
    1 from item in source.AsParallel()
    2 where Compute(item)
    3 select item;
    ```
  - **Tasks**
    - Automatically scheduled on threads (customizable)
    - Can be canceled, waited on, return values, spawn sub-tasks
    ```csharp
    1 Parallel.Invoke(() => Task1(), () => Task2());
    ```
  - **Futures**
    - Value property will block until computation completes
    ```csharp
    1 var result = Future.StartNew<int>(() => ComputeValue());
    2 Console.WriteLine(result.Value);
    ```

- Thunks
Other Languages

- Supported by Microsoft
  - C#, VB.NET, Managed C++, Managed JScript
  - IronPython, IronRuby
    - Can use subset of language’s standard libraries
  - F# - functional language based on ML, Ocaml
  - Powershell - object-oriented (vs. text) command shell

- 3rd Party
  - Boo – Python-inspired, powerful meta-programming
  - IronScheme – R6RS compliant, CLI integration
  - LSL – Scripting in Second Life (using Mono)
  - BrainF**K, LOLCode – Why not?
Name That Language!

1. `Console.WriteLine("Hello World");`
2. `puts "Hello World"`
3. `Console::WriteLine("Hello World");`
4. `print "Hello World"`
5. `printfn "Hello World\n";`
6. `Console.WriteLine("Hello World")`
7. `"Hello World"`
8. `VISIBLE "HAI WORLD"`
Name That Language!

1. Console.WriteLine("Hello World");
2. puts "Hello World"
3. Console::WriteLine("Hello World");
4. print "Hello World"
5. printfn "Hello World\n";
6. Console.WriteLine("Hello World")
7. "Hello World"
8. VISIBLE "HAI WORLD"

C#  IronRuby  Managed C++  IronPython  F#  VB.NET  Powershell  LOLCode
Questions?