Scala.NET: What you can do with it today

Miguel Garcia
http://lamp.epfl.ch/~magarcia
LAMP, EPFL

2011-06-03

Outline

1. Scala.NET in 30 seconds
2. Use cases
   - Demo: Compiling and debugging .NET Framework apps
   - Migrating from JDK to .NET
   - Write once, run cross-platform
3. Scala.NET features
   - Already available
   - Work in progress
4. Growing ecosystem
   - What works now
   - Contributions welcome

The Scala compiler is modular:

- frontend and backend are replaceable components.

Scala.NET instantiates that architecture for .NET:

- *(frontend, backend)* read and write assemblies.
- In between: type system and AST rewritings.
  - They are the same for all platforms.
  - Therefore: *same language semantics across platforms*.

It bootstraps (`scalacompiler.exe` compiles its own sources)

Demo: Compiling and debugging .NET Framework apps

```
import System.Collections.IList

object CountAll {

  def doCount(sample: Int, is: IList) = {
    val enu = is.
    var count =
    while(enu.Mo
      if(enu.Cur
    }
    count
  }

}

object Main {
  def main(args:
    val ilist = new System.Collections.ArrayList()
    ilist.Add(1); ilist.Add(2);
    scala.Console.println(CountAll.doCount(1, ilist))
  }
}
```

```
GetEnumerator                    IEnumerator
!=(: Any)                        Boolean
##                               Int
+(other: String)                 String
->[B](y: B)                      (IList, B)
==(: Any)                        Boolean
Add(value: AnyRef)               Int
asInstanceOf[T]                  T
Clear                            Unit
clone()                          AnyRef
Contains(value: AnyRef)          Boolean
```

Migrating from JDK to .NET in two easy steps:

1. convert sources with `jdk2ikvm`[1]

```
1 object HelloWorld {
2   val x = (new String(Array('h','e','l','l','o'))
3             indexOf 2)
4 }
5
6
7
```

```
1 object HelloWorld {
2   val x = (java.lang.String.instancehelper_indexOf(
3               java.lang.String.newhelper(
4                 Array('h','e','l','l','o')) , 2)
5             )
6 }
7
```

2. recompile
   - just like any other app
   - Scala.NET does not special-case for IKVM in any way

[1] `http://lamp.epfl.ch/~magarcia/jdk2ikvm`

Write once, run cross-platform

> *Scala programs that use only the Scala Library compile unmodified on both JVM and .NET*
>
> *Scala Library == Cross-platform SDK*

- Prefer `scala.io` over `java.io` or `System.IO`
- Other examples: Parallel Collections, Scala Reflection, etc.

A sidenote. `jdk2ikvm` uses an *underexploited* trick of the Scala compiler: *source-to-source conversions*

```
        x1 = "abc".substring(0, AboutPositions.this.padding)
app          |-------------------------------------------|[106:153]
fun          |--------------|                             [106:121]
quali        |----|                                       [106:111]
arg0                        ||                           [122:123]
arg1                           |-------------------------| [125:152]
```

- Pros:
  - Build your own Scala pre-processor for type-directed transforms, code expansion, etc.
- Cons:
  - Additional build step
  - Type checking performed twice (the pre-processor works on fully-typed ASTs)

Scala.NET features:

1. `scalalib.dll` was migrated via `jdk2ikvm` and thus requires the IKVM runtime libraries.

2. The same best practice:

   Don't expose JDK dependencies
   in public API of the Scala SDK

   keeps IKVM as internal implementation aspect.

3. More functionality in the Scala SDK
   means more cross-platform Nirvana.

Work in progress

Currently, the backend erases type params and args ("generics") as when emitting Java bytecode.

- not nice with other languages, we miss on their generic APIs.
- we'll throw a party after closing that bug.

After that:

- Visual Studio plugin:
  - MVC architecture.
  - Platform-agnostic *Model* (aka *presentation compiler*)
  - *GUI plumbing* TBD.
- Emitting binary assemblies.
  - Off-the-shelf solutions: *CCI*, *IKVM.Reflection*.

Growing ecosystem: Other stuff already working

Code that builds upon the *core compiler infrastructure* works fine with Scala.NET:

- continuations, virtualized Scala
- Type Debugger
- Scala Integrated Query, to name just one staged DSL.

Fine print:

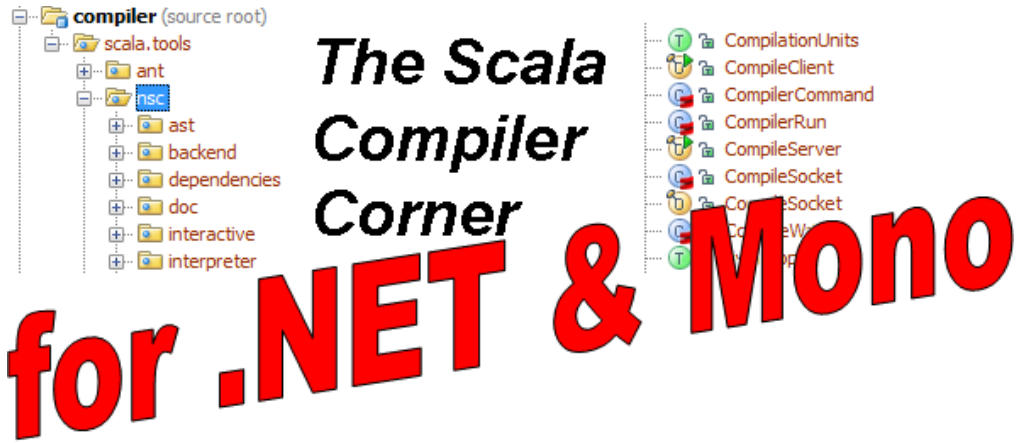
- Their *logic* requires no change. JDK-based implementation?
  - Run `jdk2ikvm` on them, or update to Scala SDK.
- compiler plugins can be loaded as `dlls`[2].

[2] Sec. 4 in `http://lamp.epfl.ch/~magarcia/ScalaNET/2011Q1/NotesAboutCustomMods.pdf`

Growing ecosystem: Contributions welcome

1. REPL for Scala.NET
2. Extending Scaladoc to emit API docs following .NET format.
3. field testing `jdk2ikvm` on apps in the "Scala Corpus"[3]

Time to bookmark



`http://lamp.epfl.ch/~magarcia/ScalaNET`

[3] `http://github.com/alacscala/scala-corpus`