
JavaOne™

ORACLE

The Not Java That's Not Scala: Alternatives for EE Development

Justin Lee, Oracle

Introduction

- Who am I?
- Using Java since 1996
- GlassFish and Grizzly team member
  - websockets
- Basement Coder
- Used various languages over the years
  - python, c, c++, scala, ...

Introduction

- Who am I not?
- A Scala hater
- A language geek
- Compiler jockey

Why Not Java?

- Java is 15+ years old now
  - Long time since the last release
  - Pace of change is geological
- Verbosity
- The shine has worn off
- It's not cool!
- It's “enterprisey”
- it doesn't have my favorite feature!

Why Not Scala?

- Lots of advanced features that are not readily accessible to “Average Joes”
- “Academics are driving the bus” - James Gosling
- Complexity - real and perceived
- new fangled features
- esoteric features
- rampant feature abuse: *trait **->**->** {implicit def **->**->**[A, F[_, _], B](a: F[A, B]): *->*->*[A, F, B] = new *->*->*[A, F, B] {val value = a}}*

So What's Missing?

- Closures
- Syntax Cleanups
- Succinctness
  - Typing/Inference
- Generics
- Concurrency

So What's Missing?

- Functional programming
- Mutability
- Null Handling
- Extendability
  - Extension methods
  - Mixins
- Modularity

So … what would you suggest?

- Suggest is a strong word but...
- Fantom
- Gosu
- New to me, too, to one degree or another

Fantom (Formerly known as Fan)

- http://fantom.org
- Familiar Syntax
- Functional
- Static and Dynamic Typing
- “Modern” concurrency facilities
- Portable
  - compile to JavaScript and SWT both

Fantom - Portability

- Fcode
- JVM (of course)
- .Net (partial)
- JavaScript

Fantom

- Literals
  - maps: [1:"one", 2:"two"] , empty [:]
  - lists: [0, 1, 2], empty [,]
  - URIs: `/dir/file.txt`
  - Types: Int#
  - Slots: Int#plus

Fantom

```
public class Person {
  private String name;
  private int age;
  public Person(String s, int x) { name = s; age = x; }
  public String getName() { return name; }
  public void setName(String x) { name = x; }
  public int getAge() { return age; }
  public void setAge(int x) { checkAge(age); age = x; }
  int yearsToRetirement(int retire) { return (retire == -1 ? 65 : retire) - age }
}
```

Fantom

```
class Person {
  Str name
  new make(String s, Int x) { name = s; age = x; }
  Int age { set { checkAge(it); &age = it }
  Int yearsToRetirement(Int retire := 65) { return retire - age }
}
```

Fantom

Inheritance and Mixins:

```
class Foo {
  virtual Void bar() { echo("foo") }
}
mixin Bar {
   abstract Int baz
   Void yep(Int x) { baz = x }
}
```

```
class Bob : Foo, Bar {
   override baz := 10
  override Void bar() { echo("bob") }
}
```

JavaOne ORACLE

Fantom

Closures

```
["red", "yellow", "orange"].each |Str color| { echo(color) }

10.times |Int i| { echo(i) }

10.times |i| { echo(i) }

10.times { echo(it) }

files = files.sort |a, b| { a.modified <=> b.modified }
```

Functions

```
add := |Int a, Int b->Int| { return a + b }

nine := add(4, 5)
```

Fantom

Dynamic typing

```
obj->foo        // obj.trap("foo", [,])
```

Nullable Types

```
Str   // never stores null

Str?  // might store null
```

Fantom

Immutability

```
const class Point{
  new make(Int x, Int y) { this.x = x; this.y = y }
  const Int x
  const Int y
}
const static Point origin := Point(0, 0)
stooges := [“Moe”,”Larry”,”Curly”].toImmutable
```

Fantom

Concurrency

```
// spawn actor which aynchronously increments an Int msg
actor := Actor(group) |Int msg->Int| { msg + 1 }

// send some messages to the actor and block for result
5.times echo(actor.send(it).get)
```

Fantom

Standard Libraries

```
// get file over HTTP

WebClient(`http://fantom.org`).getStr

// get tomorrow's date

Date.today + 1day  // date literals

// compute HTTP Basic auth string

res.headers["Authorization"] = "Basic " + "$user:$pass".toBuf.toBase64
```

Fantom

Modularity and Meta-Programming

```
// find pods available for installation in project "Cool Proj"
repo := Repo.makeForUri(`http://my-fantom-repo/`)
pods := repo.query(Str<|"* proj.name=="Cool Proj"|>)
pods.each |p| { echo("$p.name $p.version $p.depends") }
```

Fantom

Modularity and Meta-Programming

```
// find all types installed in system which subclass Widget
Pod.list.each |pod| {
  pod.types.each |type| {
    if (type.fits(Widget#)) echo(type.qname)
  }
}
```

Fantom

Modularity and Meta-Programming

```
// find pods available for installation in project "Cool Proj"
repo := Repo.makeForUri(`http://my-fantom-repo/`)
pods := repo.query(Str<|"* proj.name=="Cool Proj"|>)
pods.each |p| { echo("$p.name $p.version $p.depends") }
```

Fantom

Modularity and Meta-Programming

```
// find pods available for installation in project "Cool Proj"
repo := Repo.makeForUri(`http://my-fantom-repo/`)
pods := repo.query(Str<|"* proj.name=="Cool Proj"|>)
pods.each |p| { echo("$p.name $p.version $p.depends") }
```

Fantom

DSLs

- embed other languages in your fantom code
- similar to CDATA sections in XML
- AnchorType <|...|>
- echo(Str <|now is the time for all good men
  to come to the aid of their country.|>)
- Regex <|^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$|>

Fantom

```
class RegexDslPlugin : DslPlugin {
  new make(Compiler c) : super(c) {}
  override Expr compile(DslExpr dsl)  {
    regexType := ns.resolveType("sys::Regex")
    fromStr := regexType.method("fromStr")
    args := [Expr.makeForLiteral(dsl.loc, ns, dsl.src)]
    return CallExpr.makeWithMethod(dsl.loc, null, fromStr, args)
  }
}
// register in indexed props in build script
index = ["compiler.dsl.sys::Regex": "compiler::RegexDslPlugin"]
```

Fantom - Portability

```
@Js
class Demo {
  Void main() {
    // dumps to your browser's JS console
    10.times |x| { echo("x: $x")
  }
}
```

Fantom - Tales

```
@Js class ClickClickJs {
  Void main() {
    Jq.ready {
      Jq("#click").click|cur, event| {
        Win.cur.alert("Click Click")
        event.preventDefault
      }
    }
  }
}
```

```
class Routes{
  Route[] routes := Route[
    Route{map="/clickclick";
to="ClickClick";}
  ]
}
```

Fantom - Draft

```
const class MyMod : DraftMod {
  new make() {
    pubDir = `...`.toFile
    router = Router {
      routes = [ Route("/", "GET", #index),
        Route("/echo/{name}/{age}", "GET", #echo)      ]
    }
  }
```

Fantom - Draft

```
Void index() {
  res.headers["Content-Type"] = "text/plain"
  res.out.printLine("Hi there!")
}
Void echo(Str:Str args) {
  name := args["name"]
  age  := args["age"].toInt
  res.headers["Content-Type"] = "text/plain"
  res.out.printLine("Hi $name, you are $age years old!")
}
```

Fantom - Resources

- http://fantom.org #fantom on irc.freenode.net
- http://www.talesframework.org/
- Draft Mini Web Framework https://bitbucket.org/afrankvt/draft/wiki/Home
- http://spectreframework.org/
- Twitter
  - @afrankvt
  - @briansfrank

Gosu

- http://gosu-lang.org
- Again, familiar syntax
- Statically typed
- Enhancements
- Closures
- Simplified generics
- ***Open Type System***

Gosu

- == Object equality
- === Instance equality
- >, <, etc. works with java.lang.Comparable
- Standard logical operators
- Null Safety

```
print( x?.length ) // prints "null"
```

- `var zeroToTenInclusive = 0..10`

Gosu - Properties!

```
public class MyJavaPersonClass {
  String _name;

  public String getName() {
    return _name;
  }

  public void setName(String s) {
    _name = s;
  }
}
```

```
var p = new MyJavaPersonClass()
p.Name = "Joe"
print( "The name of this person is ${p.Name}")
```

Gosu

```
uses java.util.List
class SampleClass {
  var _names : List<String> // a private class variable, which is a list of Strings

  construct( names : List<String> ) { _names = names }

  function printNames( prefix : String ) {
    for( n in _names ) { print( prefix + n ) }
  }

  property get Names() : List<String> { return _names }
}
```

Gosu

```
var c = new SampleClass({"joe", "john", "jack"})
c.printNames("* ")
* joe
* john
* jack

print( c.Names )
[joe, john, jack]
```

Gosu

```
uses java.util.List
class SampleClass {
  var _names : List<String>

  construct( names : List<String> ) { _names = names }

  function printNames( prefix : String ) {
    for( n in _names ) { print( prefix + n ) }
  }

  property get Names() : List<String> { return _names }
}
```

Gosu

```
uses java.util.List
class SampleClass {
  var _names : List<String> as Names

  construct( names : List<String> ) { _names = names }

  function printNames( prefix : String ) {
    for( n in _names ) { print( prefix + n ) }
  }
}
```

Gosu

```
uses java.util.List
class SampleClass {
  var _names : List<String> as readonly Names

  construct( names : List<String> ) { _names = names }

  function printNames( prefix : String) {
    for( n in _names ) { print( prefix + n ) }
  }
}
```

Gosu

```
uses java.util.List
class SampleClass {
  var _names : List<String> as readonly Names

  construct( names : List<String> ) { _names = names }

  function printNames( prefix : String  = "> ") {
    for( n in _names ) { print( prefix + n ) }
  }
}
```

Gosu

```
class MyRunnable implements Runnable {
  delegate _runnable represents Runnable

  construct() {
    _runnable = new Runnable() {
      override function run() { print("Hello, Delegation") }
    }
  }
  property get Impl() : Runnable { return _runnable }
  property set Impl( r : Runnable ) { _runnable = r }
}
```

Gosu

```
class MyRunnable implements Runnable {
  delegate _runnable represents Runnable

  construct() {
    _runnable = new Runnable() {
      override function run() { print("Hello, Delegation") }
    }
  }
  property get Impl() : Runnable { return _runnable }
  property set Impl( r : Runnable ) { _runnable = r }
}

var x = new MyRunnable()
x.Impl = new Runnable() {
  override function run() { print("Hello, Delegation 2") }
}
```

Gosu - Donkey Patching

Extensions

```
enhancement MyStringEnhancement : String {
  function printMe() {
    print( this )
  }
}

"Hello Enhancements".printMe()
```

Enhancements are statically dispatched which means they cannot be used to implement interfaces or to achieve polymorphism

Gosu

Enhancements

- String: rightPad(), leftPad(), center(), toDate(), toBoolean()
- File: read(), write(), eachLine()

Closures (Blocks)

```
var listOfStrings = {"This", "is", "a", "list"}
var longStrings = listOfStrings.where(\ s -> s.length>2)
print( longStrings.join(", ") )


var r : Runnable
r = \-> print("This block was converted to a Runnable")
```

Gosu

Now for the fun part.

Gosu

Now for the fun part.

XML marshalling!

Gosu

Now for the fun part.

XML marshalling!

JAXB is “great” and all but it can be … painful.

Gosu

Now for the fun part.

XML marshalling!

JAXB is “great” and all but it can be … painful.

Gosu has great alternative.

Gosu

Now for the fun part.

XML marshalling!

JAXB is “great” and all but it can be … painful.

Gosu has great alternative.

But first <shudder/> an XSD...

Gosu

```
<?xml version="1.0"?>
 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Person">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="firstname" type="xs:string"/>
     <xs:element name="lastname" type="xs:string"/>
     <xs:element name="age" type="xs:int"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 </xs:schema>
```

Gosu

```
var bob = new myapp.example.Person()
bob.Age = 32
bob.Firstname = "Bob"
bob.Lastname = "Loblaw"
bob.print()
```

Gosu

```
var bob = new myapp.example.Person()
bob.Age = 32
bob.Firstname = "Bob"
bob.Lastname = "Loblaw"
bob.print()
```

What about WSDL?

Gosu

```
var bob = new myapp.example.Person()
bob.Age = 32
bob.Firstname = "Bob"
bob.Lastname = "Loblaw"
bob.print()

//  weather.wsdl
var x = new myapp.weather()
var forecast = x.GetCityForecastByZIP("95816")
print( forecast.ForecastResult.Forecast.map( \ f -> f.Description ).join("\n") )
```

Gosu

Feature Literals

```
var sub = String#substring(int)  // class reference
print( sub.invoke( "Now is the time for all good men", 2 ) )


// instance reference
    var sub = "Now is the time for all good men"#substring(int)
    print( sub.invoke( 2 ) )
```

Gosu

Feature Literals

```
var sub = String#substring(int)  // class reference
print( sub.invoke( "Now is the time for all good men", 2 ) )


// instance reference
    var sub = "Now is the time for all good men"#substring(2)
    print( sub.invoke() )
```

Gosu

Feature Literals

```
var sub = String#substring(int)  // class reference
print( sub.invoke( "Now is the time for all good men", 2 ) )


// instance reference
    var sub = "Now is the time for all good men"#substring(2)
    print( sub.invoke() )

var component = new Component(foo#Bar#Bob)
```

Gosu - Ronin

```
<%@ extends ronin.RoninTemplate %>
<%@ params(name : String) %>
<html>
    <body>
        Hello ${name}!
    </body>
</html>
```

Gosu - Ronin

```
package controller
    uses ronin.*
    class Main extends RoninController {
      function hello(name : String) {
          view.Hello.render(writer, name)
    }
}
```

Gosu

- http://www.gosu-lang.org
- Web framework http://ronin-web.org/
- ORM: http://ronin-web.org/Tosa.html
- Build tool: Aardvark http://vark.github.com/
- JVM Language Summit: http://medianetwork.oracle.com/media/show/17005
- Twitter: @carson_gross
- IRC: #gosulang @ irc.freenode.net

ORACLE

Justin Lee, Oracle
http://antwerkz.com
@evanchooly
http://github.com/evanchooly
http://www.basementcoders.com