Absorbing Scala in the Java Ecosystem

Eishay Smith
kaChing

kaChing

Who Am I

- Director of Engineering at kaChing - Rocking the inventing world !
- Principal Engineer 

- IBM Research

Blogging http://www.eishay.com

On the menu

- Getting started
- Scala and Java
- Runtime
- Build
- IDEs
- Test
- People

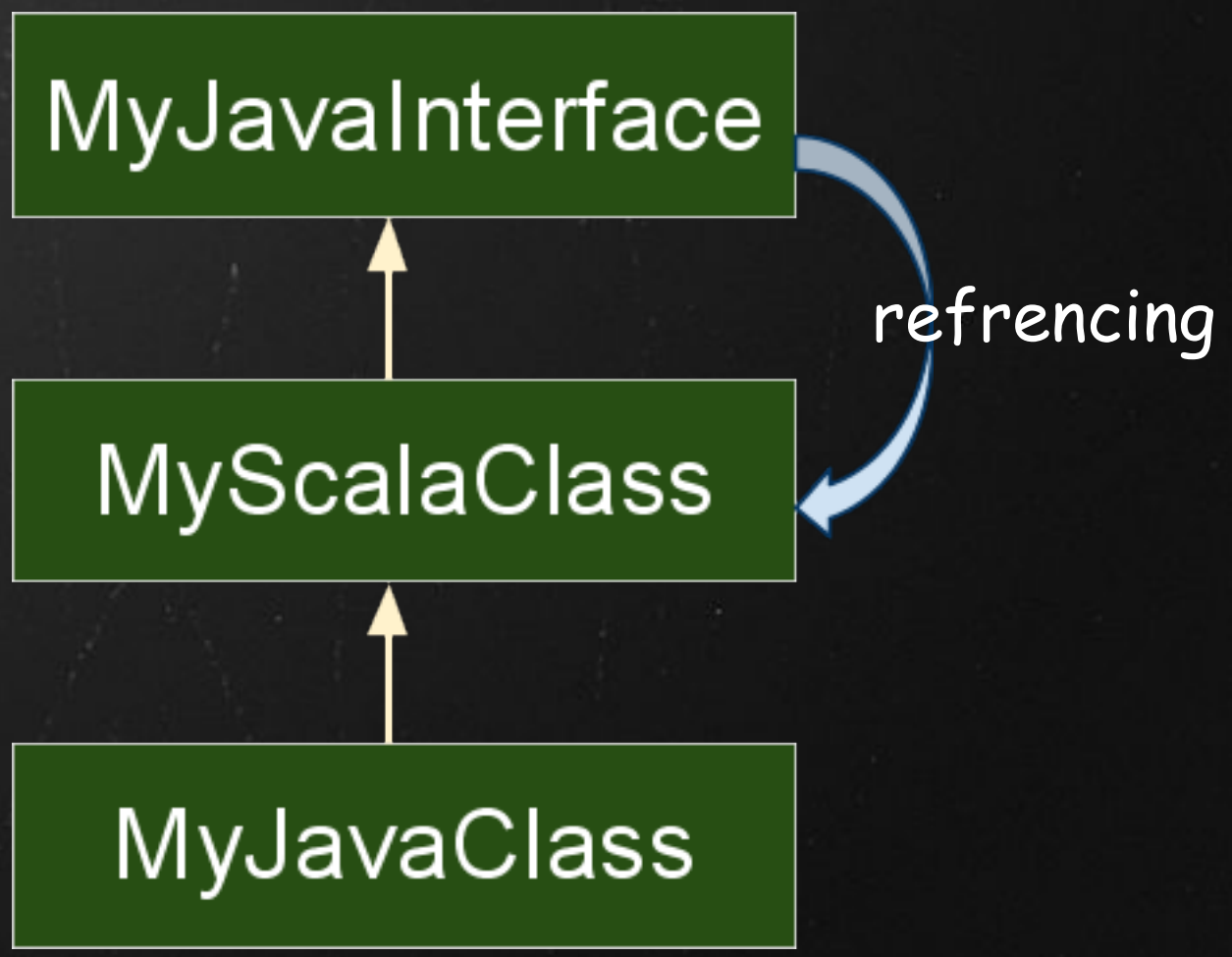
Scala and Java

- Different syntax and philosophy
  - Unlike the Groovy you can't write Java syntax in Scala
- Better interoperability
- Compiles to the same classes and interfaces
- Type safe
  - Stricter then Java
  - Don't get in your way
- Reduces boilerplates
  - Closures, implicits, meta-programming
- Easily sharing libraries, both ways
- Profiling, JMX, Serialization ....

The Scala Runtime

- Behave the same on runtime
  - Your SysAdmin/Ops won't tell the diferance
- As fast as Java
  - Static types, no introspection (see dynamic languages)
  - Using primitives (though they look like objects)
  - Better concurrency patterns (see Actors)
- Taking advantage of the JIT
- Makes the GC happy (immutability FTW)

Cross language dependencies



Build

Scalac knows how to read java code ! Needed for circular dependencies.

```
<target name="compile" depends="copyjars"
  description="Compiles Java source and copies other source files to the WAR.">
  <mkdir dir="${dstdir}" />
  <copy todir="${dstdir}">
    <fileset dir="${srcdir}">
      <exclude name="${javafiles}" />
      <exclude name="${scalafiles}" />
    </fileset>
  </copy>
  <scalac
    destdir="${dstdir}"
    scalacdebugging="yes">
    <src path="${srcdir}"/>
    <classpath refid="project.classpath"/>
  </scalac>

  <javac srcdir="${srcdir}"
          destdir="${dstdir}"
          classpathref="project.classpath"
          source="1.5"
          target="1.5"
          nowarn="true"
          debuglevel="lines,vars,source"
          debug="on" />

  <scalac
    destdir="${dstdir}"
    scalacdebugging="yes">
    <src path="${testdir}"/>
    <classpath refid="project.classpath"/>
  </scalac>
  <javac srcdir="${testdir}"
          destdir="${dstdir}"
          classpathref="project.classpath"
          debug="on" />
</target>
```

Testing

- Scala has some great Testing frameworks
  - Specs, ScalaTest
  - They do integrate nicely with existing frameworks
- Can use bare bones JUnit (any version)
- ANT JUnit plugin does not work with Scala Sources
  - But does work with class files
- JUnit may use static suite() methods in the test class
  - Problem: no static methods in Scala
  - There is a solution

JUnit, ANT & Scala

Instead of

```
<attribute name="file-pattern" default="**/Test*.java,
➲ **/Test*.scala"/>
<junit fork="..." forkmode="..." dir="...">
  ...
  <batchtest fork="..." todir="...">
    ...
    <fileset dir="@{test-src-dir}"
    ➲ includes="${test.package.path}@{file-pattern}"/>
  </batchtest>
</junit>
```

Use

```
<attribute name="file-pattern" default="**/Test*.class"/>
<attribute name="excludes" default="**/*$*.class"/>
<junit fork="..." forkmode="..." dir="...">
  ...
  <batchtest fork="..." todir="...">
    ...
    <fileset dir="@{test-build-dir}"
    ➲ includes="${test.package.path}@{file-pattern}"
      excludes="@{excludes}"/>
  </batchtest>
</junit>
```

Getting around static methods

```
package test1
object Test{
  def scalaStatic = "scala static"
}
```

```
javap bin/test1/Test
Compiled from "Test.scala"
public final class test1.Test extends java.lang.Object{
    public static final java.lang.String scalaStatic();
    public static final int $tag()       throws
    ➲ java.rmi.RemoteException;
}
javap bin/test1/Test$
Compiled from "Test.scala"
public final class test1.Test$ extends java.lang.Object implements
➲ scala.ScalaObject{
    public static final test1.Test$ MODULE$;
    public static {};
    public test1.Test$();
    public java.lang.String scalaStatic();
    public int $tag()       throws java.rmi.RemoteException;
}
```

```
package test1.java;
import test1.Test;
public class JavaTest{
  public String usingScalaStatic() {return Test.scalaStatic();}
}
```

Your organization

- Integrating smartly with current Java project
  - Use one way dependency
- Keeping the Scala illiterate IDEs happy
  - Not everyone will jump on the wagon
  - Take advantage on your IoC framework
    - Spring
  - Use Java interfaces to make the IDE happy

IDEs

- Good Support by the three big (and free) IDEs out there
  - Eclipse, NetBeans and IDEA
- Healthy competition
- Soon to come: better IDE support
  - Martin Odersky is working on improving the IDE support with better compiler infrastructure
- Scala plugin does not come with the IDE
  - Other team members may not install it

Get people interested

- Host a Scala BASE
- Invite Scala Speakers
- Lunchtime talks

TShirts

(Scala /: LinkedIn) (_+_)

Getting started

- Scala is more then just nice syntax
- You may start writing Java code in Scala
- Start with testing
  - Not with production code
  - Test Java code from Scala
  - Once way dependency
- Absorb Scala slowly, don't get drunk !
  - Your first Scala code will not be perfect
  - It takes a while to understand the Scala philosophy

kaChing and Scala

- New ways of thinking
- Query engine - the kaChing revolutionary service container
  - Written by the students of Martin Odersky
  - Java based
  - Functional
  - To be open sourced
- Blending Java and Scala

... hiring

kaChing