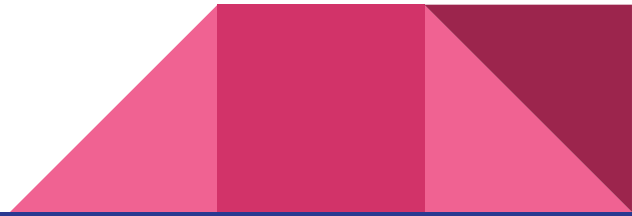
You can run (almost) anything on the JVM, so why not run everything on the same JVM?! The magic of interoperability!

Anne DeCusatis, May 2019

What is the JVM?

- The Java Virtual Machine (JVM) was created as a cross-platform environment to run code from the programming language Java
- As long as the language can compile to Java bytecode, the source language doesn’t have to be Java



Scala

Scala runs in SBT on the JVM

```scala
package interop.example1

object ScalaMain {

  def main(args: Array[String]): Unit = {
      println("Initializing Scala!")
      println("Exiting Scala!")
  }

}
```

```
sbt:interop> runMain interop.example1.ScalaMain
...
[info] Running interop.example1.ScalaMain
Initializing Scala!
Exiting Scala!
[success] Total time: 3 s
```

Java

Scala interoperates with Java

```java
package interop.example1;

public class JavaMain {
  public static void main(String[] args) {
      System.out.println("Initializing Java!");
      ScalaMain$.MODULE$.main(args);
      System.out.println("Exiting Java!");
  }
}
```

```scala
package interop.example1

object ScalaMain {
  def main(args: Array[String]): Unit = {
      println("Initializing Scala!")
      println("Exiting Scala!")
  }
}
```

```
sbt:interop> runMain interop.example1.JavaMain
...
[info] Running interop.example1.JavaMain
Initializing Java!
Initializing Scala!
Exiting Scala!
Exiting Java!
[success] Total time: 4 s
```

Scala interoperates with Java

```java
package interop.example1;

public class JavaMain {
  public static void main(String[] args) {
      System.out.println("Initializing Java!");
      ScalaMain$.MODULE$.main(args);
      System.out.println("Exiting Java!");
  }
}
```

```scala
package interop.example1

object ScalaMain {
  def main(args: Array[String]): Unit = {
      println("Initializing Scala!")
      println("Exiting Scala!")
  }
}
```

```
sbt:interop> runMain interop.example1.JavaMain
...
[info] Running interop.example1.JavaMain
Initializing Java!
Initializing Scala!
Exiting Scala!
Exiting Java!
[success] Total time: 4 s
```

Objects don’t exist as a Java construct, so Scala uses $.MODULE$

Python

Python also runs on the JVM

```
from interop.example1.JavaMain import main as jmain

def main():
      print("Initializing python!")
      jmain([])
      print("Exiting python!")

if __name__ == "__main__":
      main()
```

```
addSbtPlugin("info.hupel" % "sbt-jsr223" % "0.1.1")
addSbtPlugin("info.hupel" % "sbt-jython" % "0.1.1")
enablePlugins(JythonPlugin)
```

```
sbt:interop> runMain PythonMain$py
...
[info] Running PythonMain$py
Initializing python!
Initializing Java!
Initializing Scala!
Exiting Scala!
Exiting Java!
Exiting python!
[success] Total time: 2 s
```

Python also runs on the JVM

```
from interop.example1.JavaMain import main as jmain

def main():
      print("Initializing python!")
      jmain([])
      print("Exiting python!")

if __name__ == "__main__":
      main()
```

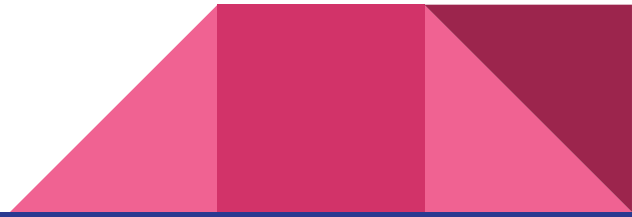
```
sbt:interop> runMain PythonMain$py
...
[info] Running PythonMain$py
Initializing python!
Initializing Java!
Initializing Scala!
Exiting Scala!
Exiting Java!
Exiting python!
[success] Total time: 2 s
```

Python also uses $ to identify internal constructs used when compiling bytecode

Clojure

Clojure Hello World

```
(ns interop.example1.core)

(defn -main
  "I can say 'Hello World'."
  ([] (println "Initializing Clojure!\nExiting Clojure!")))
```



Clojure Hello World from Scala

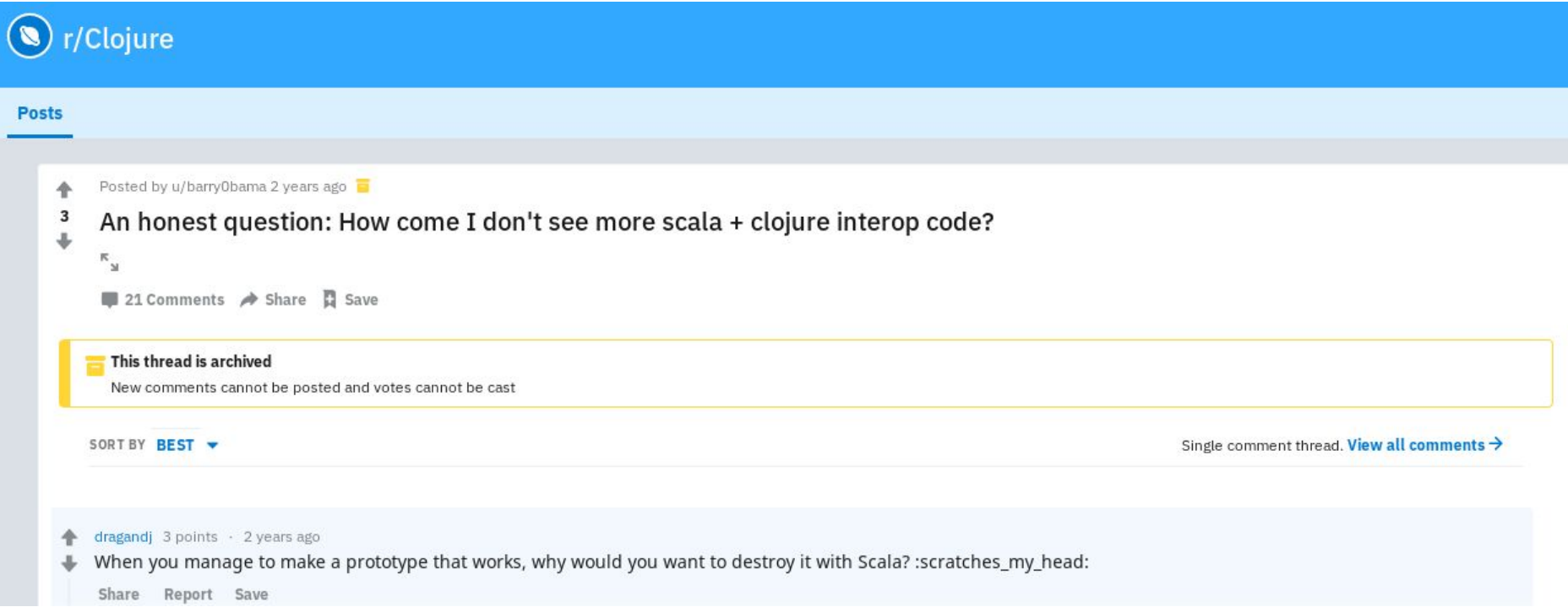
```
package interop.example1

import clojure.java.api.Clojure

object ScalaMain {
  def main(args: Array[String]): Unit = {
      println("Initializing Scala!")
      // call clojure
      // https://blog.michielborkent.nl/2016/07/26/clojure-from-scala/
      // first initialize ability to require things
      val core = Clojure.`var`("clojure.core", "require")
      // then require my namespace
      core.invoke(Clojure.read("interop.example1.core"))
      // then find my function
      val clojureMain = Clojure.`var`("interop.example1.core", "-main")
      // then run the function
      clojureMain.invoke()
      println("Exiting Scala!")
  }
}
```

```
[info] Running PythonMain$py
Initializing Python!
Initializing Java!
Initializing Scala!
[error] (run-main-0) Traceback (most recent call last):
[error]   File "PythonMain$py", line 12, in <module>
[error]   File "PythonMain$py", line 8, in main
[error]         at clojure.lang.RT.load(RT.java:466)
[error]         at clojure.lang.RT.load(RT.java:428)
[error]         at clojure.core$load$fn__6824.invoke(core.clj:6126)
[error]         at clojure.core$load.invokeStatic(core.clj:6125)
[error]         at clojure.core$load.doInvoke(core.clj:6109)
[error]         at clojure.lang.RestFn.invoke(RestFn.java:408)
[error]         at clojure.core$load_one.invokeStatic(core.clj:5908)
[error]         at clojure.core$load_one.invoke(core.clj:5903)
[error]         at clojure.core$load_lib$fn__6765.invoke(core.clj:5948)
[error]         at clojure.core$load_lib.invokeStatic(core.clj:5947)
[error]         at clojure.core$load_lib.doInvoke(core.clj:5928)
[error]         at clojure.lang.RestFn.applyTo(RestFn.java:142)
[error]         at clojure.core$apply.invokeStatic(core.clj:667)
[error]         at clojure.core$load_libs.invokeStatic(core.clj:5985)
[error]         at clojure.core$load_libs.doInvoke(core.clj:5969)
[error]         at clojure.lang.RestFn.applyTo(RestFn.java:137)
[error]         at clojure.core$apply.invokeStatic(core.clj:667)
[error]         at clojure.core$require.invokeStatic(core.clj:6007)
[error]         at clojure.core$require.doInvoke(core.clj:6007)
```

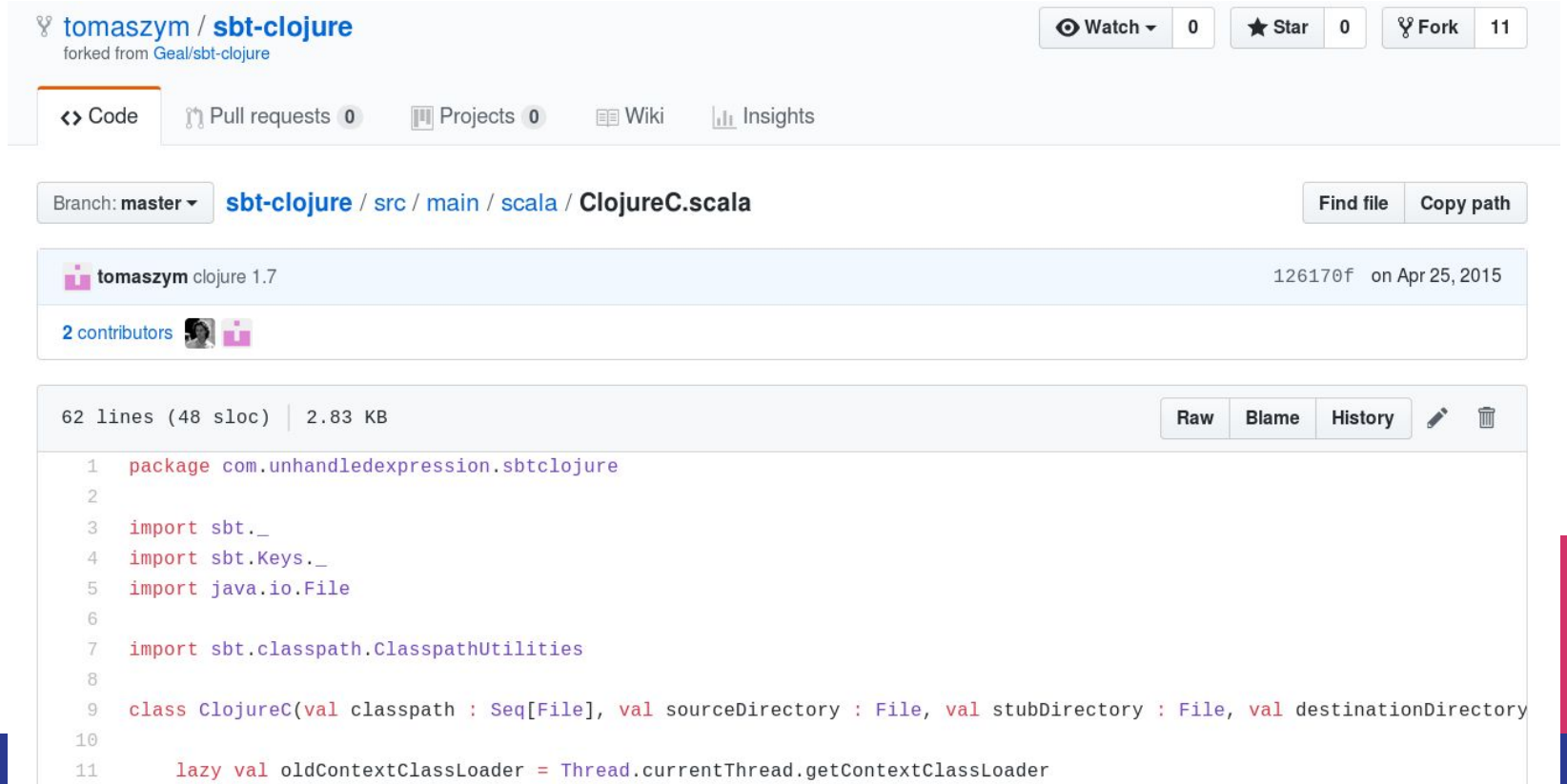
Clojure runtime exception

Eventually caused by…

```
[error] java.io.FileNotFoundException: java.io.FileNotFoundException: Could not locate
interop/example1/core__init.class, interop/example1/core.clj or interop/example1/core.cljc on classpath.
```


r/Clojure
Posts
Posted by u/barry0bama 2 years ago
3
An honest question: How come I don't see more scala + clojure interop code?
21 Comments
Share
Save
This thread is archived
New comments cannot be posted and votes cannot be cast
SORT BY BEST
Single comment thread. View all comments
dragandj 3 points · 2 years ago
When you manage to make a prototype that works, why would you want to destroy it with Scala? :scratches_my_head:
Share
Report
Save

I need an SBT plugin for Clojure



tomaszym / **sbt-clojure**
forked from Geal/sbt-clojure

Watch 0 | Star 0 | Fork 11

Code | Pull requests 0 | Projects 0 | Wiki | Insights

Branch: **master** — **sbt-clojure** / src / main / scala / **ClojureC.scala** — Find file | Copy path

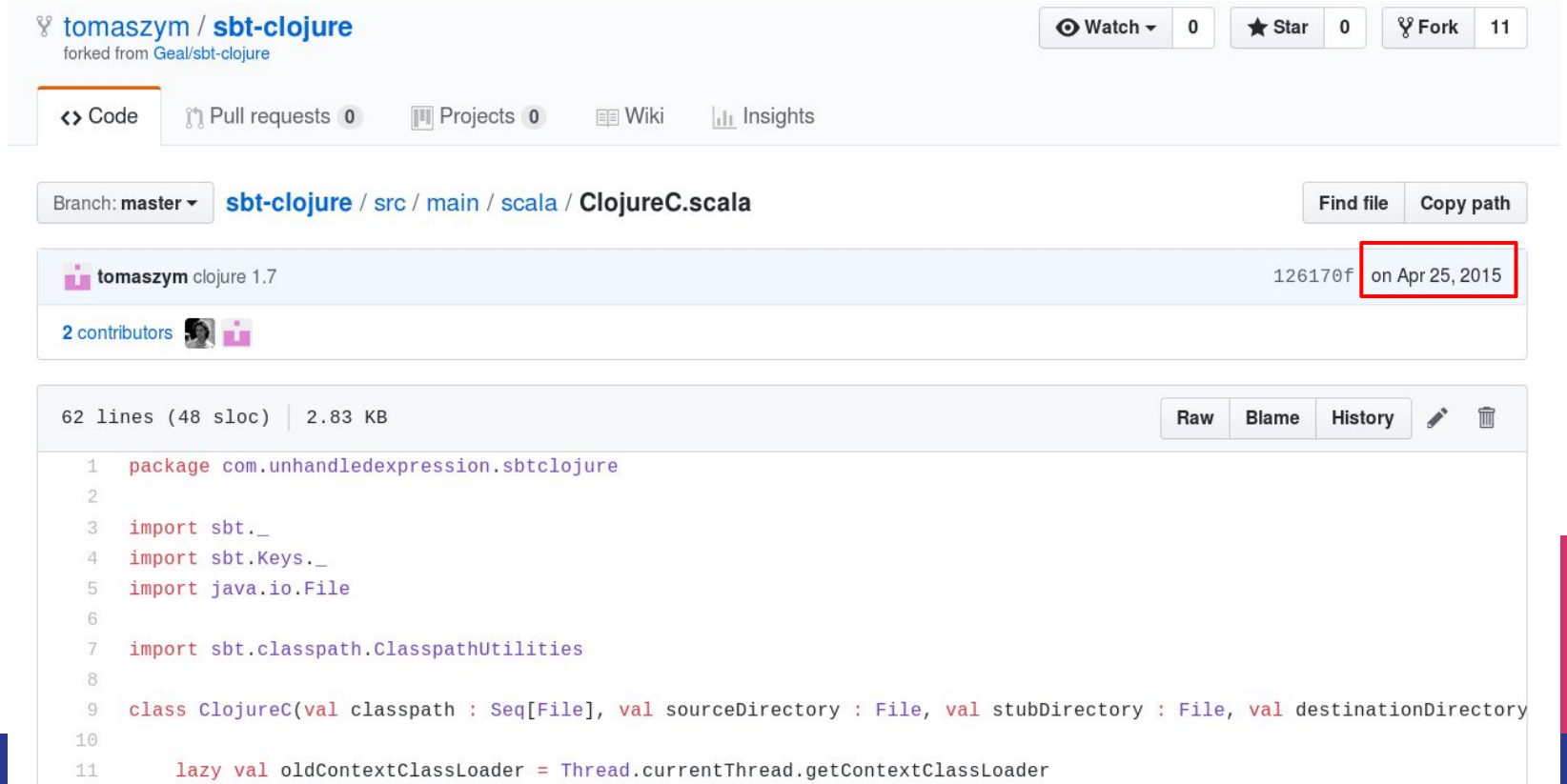
tomaszym clojure 1.7 — `126170f` on Apr 25, 2015

2 contributors

62 lines (48 sloc) | 2.83 KB — Raw | Blame | History

```
1  package com.unhandledexpression.sbtclojure
2
3  import sbt._
4  import sbt.Keys._
5  import java.io.File
6
7  import sbt.classpath.ClasspathUtilities
8
9  class ClojureC(val classpath : Seq[File], val sourceDirectory : File, val stubDirectory : File, val destinationDirectory
10
11     lazy val oldContextClassLoader = Thread.currentThread.getContextClassLoader
```

I need an SBT plugin for Clojure



Clojure

```
sbt:interop> runMain PythonMain$py
...
[info] Running PythonMain$py
Initializing Python!
Initializing Java!
Initializing Scala!
Initializing Clojure!
Exiting Clojure!
Exiting Scala!
Exiting Java!
Exiting Python!
[success] Total time: 8 s
```

Clojure

```
sbt:interop> runMain PythonMain$py
...
[info] Running PythonMain$py
Initializing Python!
Initializing Java!
Initializing Scala!
Initializing Clojure!
Exiting Clojure!
Exiting Scala!
Exiting Java!
Exiting Python!
[success] Total time: 8 s
```

Kotlin & Ruby

Additional hello world, additional AutoPlugin

```
package interop.example1

import org.jruby.Ruby

fun main() {
      println("Initializing Kotlin!")
      val rb = Ruby.getGlobalRuntime()
      rb.executeScript("puts 'Hello from an inline Ruby script!'", "RubyMain.rb")
      println("Exiting Kotlin!")
}
```

Live demo time!

Thank you!

github.com/anne-decusatis/jvm-interoperability