Scala: a Cross-Platform Language

How to build applications that span in different platform

Gianluca Aguzzi gianluca.aguzzi@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum – Università di Bologna

Talk @ Paradigmi di Progettazione e Sviluppo

22/05/2022

Contents

1 Introduction

2 Project Examples

Cross-Platform Applications

Definition

A cross-platform software consists in an application designed to work in several computing platform

- Also reffered as *platform agnostic*, *platform independent* & *multi-platform* software
- Java, per sè, could be considered as *multi-platform* language (motto: write once, run everywhere)
- Modern perspective: the same language span over several runtime & VMs (e.g. javascript, native platform, JVM, ...)
- A multi-platform application could be *polyglot*

Frameworks & Languages for Cross-Platform Development

Frameworks (UI oriented)

- Flutter: *an open source framework by Google for multi-platform native apps (starts for Android, iOS and Windows app, now support Linux, MacOS too)*
  - Motto: *Build app for any screen*
  - Pretty recent (2017)
- Xamarin: *Extension of .NET framework (tools & libraries) for supporting apps development*
- React Native: *React Native brings React's declarative UI framework to iOS and Android*
  - Motto: *Learn once, write anywhere*

Languages (Multi-platform)

- Kotlin: support several target runtime thanks to **Kotlin multiplatform** projects
  - Introduced with Kotlin 1.2 (alpha), in 1.4 becomes experimental (Still in alpha)
  - Support JS, JVM & Native platform (iOS, LLVM, Android, ...)
- **Scala** (focus of today): targets differnet platform using *compiler & sbt plugins*:
  - Scala.js: stable version to transplie Scala in JS (born in 2014!!)
  - Scala native: alpha version to support native applications

Scala Cross-Platform

Benefits

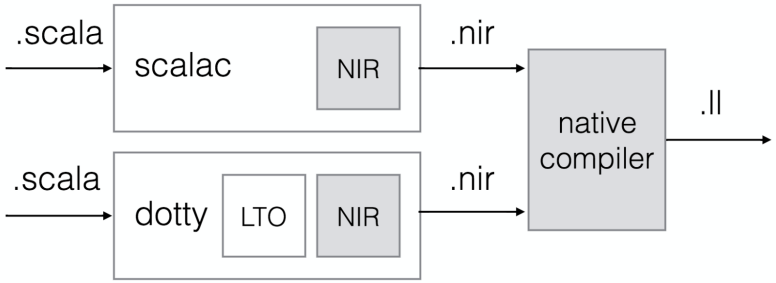
- **Shared code base**: the application logic is *shared* among several target avoiding *code duplication* & *error propagation*.
- **Full-stack oriented**: using programming language that supports *several* target enable the possibility of support the entire stack of an application (backend & client) with the same language.
- **Access to libraries & SDK**: multi-platform language could exploit libraries of a specific platform that is not intented to be used in that language (e.g. Tensorflow from Scala!!)

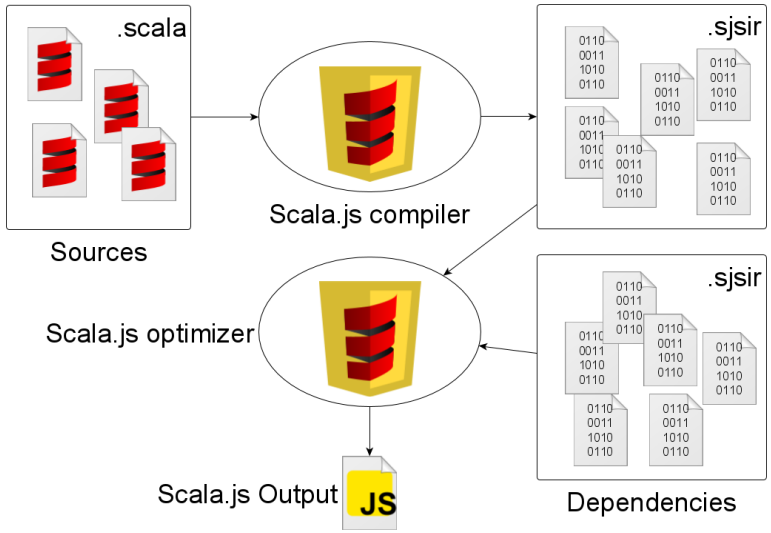
Use cases

- Cross-platform library (e.g. Cats 🔗, Monix 🔗)
- Web application development (Scala.js)
- Robotics & embeded systems (native)
- Shared code for android & iOS app
- **NB!** the use case *"target code to scala code"* is typically not considered

How Scala Native & Scala.Js work [1]

- Compiler takes plain Scala code
- Using a compiler plugin produces an intermediate representation (IR) that contain platform-dependent aspects
- Using IR the compiler processes optimization, linking, and dependency management.





[1] Sébastien Jean R Doeraene. "Cross-Platform Language Design". en. In: (2018). doi: 10.5075/EPFL-THESIS-8733. url: http://infoscience.epfl.ch/record/256862

Caveats

- In pure cross Scala project, you **can't** use JVM ecosystem (e.g. Thread?)...
  - ... unless you build ad-hoc facades
  - Several reimplemented API already exisist
- Javascript & native libraries can be used **only** in the corresponding runtime (in JVM you still cannot use JS libraries)
- For using native & JS libraries, you have to build ad-hoc facade (a là TypeScript with typings) . . .
  - For Scala.js an ScalablyTyped aims dererving Scala typing starting from TypeScript project
- Application bundle size could be large since the code should include part runtime & standard library
  - Scala.js bundle could easily reach $\sim$ 1 Mb for the output file.

Scala.js & Scala native configuration: SBT is your friend!

- Scalac does not natively produce JS & Native .class
- You have to enable *plugin* via SBT configuration

project/plugins.sbt

```
addSbtPlugin(
 "org.scala-js" % "sbt-scalajs" % "1.10.0"
)
// for native
addSbtPlugin(
 "org.scala-native" % "sbt-scala-native" % "0.4.4"
)
```

build.sbt

```
enablePlugins(ScalaJSPlugin) // or ScalaNativePlugin
scalaVersion := "3.1.2"
// for JS
scalaJSUseMainModuleInitializer := true
```

- ... still single platform

Multi Platform Setup

- Another SBT plugin to configure multiple plaform project: **sbt-crossproject**
- Enforce a project structure (configurable by different *flavour*)
  - `CrossType.Pure`: pure cross platform project (e.g. libraries), all code is placed in `/src`
  - `CrossType.Full`: project with platform specific code (e.g. UI)
    - `shared`: code purerly multi platform (e.g. data structure, interfaces, ...)
    - `js` and `jvm` and `naitve`: contain application platform specific code (e.g. library usage, GUI, ...)

project/plugins.sbt

```
val crossOrg = "org.portable-scala"
val crossNative = "sbt-scala-native-crossproject"
val crossJs = "sbt-scalajs-crossproject"
addSbtPlugin(crossOrg % crossJs % "1.1.0")
addSbtPlugin(crossOrg % crossNative % "1.1.0")
addSbtPlugin(
  "org.scala-js" % "sbt-scalajs" % "1.10.0"
)
addSbtPlugin(
  "org.scala-native" % "sbt-scala-native" % "0.4.4"
)
```

build.sbt

```
crossProject(JSPlatform, NativePlatform, JVMPlatform)
  .crossType(CrossType.Full) // or CrossType.Pure
// common settings, use %%% for cross library
  .settings(
    libraryDependencies ++= Seq(
      "org.scalatest" %%% "scalatest" % "3.2.12"
    )
  )
  .jsSettings()
  .nativeSettings()
  .jvmSettings() // standard "scala" world
```

Contents

1 Introduction

2 Project Examples

Facade for JS library

Examples I

Library Facade: Neaptic

- You want to use a library that does not exist in JVM ecosystem
- You can use unsafely (not recommand) or by *creating safe facade*

Cross Target Applicaiton: Winnig Four

- You have an application logic written in pure scala (born for JVM)
- Then you want to share it in different platform (e.g. Web, Android, ...)
  1. Convert the project to a full cross-project
  2. Mantain the core logic in the shared part
  3. Create ad-hoc GUI (or any specific platform code)

Full-Stack application: Todo

Full-Stack Application

Scala: a Cross-Platform Language

How to build applications that span in different platform

Gianluca Aguzzi gianluca.aguzzi@unibo.it

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum – Università di Bologna

Talk @ Paradigmi di Progettazione e Sviluppo

22/05/2022

References I

[1] Sébastien Jean R Doeraene. "Cross-Platform Language Design". en. In: (2018). doi: 10.5075/EPFL-THESIS-8733. url: http://infoscience.epfl.ch/record/256862.

[2] *Kotlin Multiplatform | Kotlin*. https://kotlinlang.org/docs/multiplatform.html. (Accessed on 05/22/2022).

[3] *React Native · Learn once, write anywhere*. https://reactnative.dev/. (Accessed on 05/22/2022).

[4] *Scala Native — Scala Native 0.4.5-SNAPSHOT documentation*. https://scala-native.readthedocs.io/en/latest/. (Accessed on 05/22/2022).

[5] *Scala.js*. https://www.scala-js.org/. (Accessed on 05/22/2022).

[6] *Showcase - Flutter apps in production*. https://flutter.dev/showcase. (Accessed on 05/22/2022).