Building projects with SBT

If you have reached this section you probably have a system that is now able to compile and run Scala Native programs.

Minimal sbt project

Start within a new folder, and create a file `project/plugins.sbt` as follows:

```
addSbtPlugin("org.scala-native" % "sbt-scala-native"  % "0.1.0")
```

Create a file `project/build.properties` to define the sbt version as follows:

```
sbt.version = 0.13.13
```

define a new `build.sbt`:

```
enablePlugins(ScalaNativePlugin)

scalaVersion := "2.11.8"
```

and now you can write your first application in `./src/main/scala/HelloWorld.scala`:

```
package example

object Main {
  def main(args: Array[String]): Unit =
    println("Hello, world!")
}
```

now simply run `sbt run` to get everything compiled and have the expected output!

Sbt settings and tasks

Since	Name	Type	Description
0.1	`compile`	`Analysis`	Compile Scala code to NIR
0.1	`run`	`Unit`	Compile, link and run the generated binary
0.1	`package`	`File`	Similar to standard package with addition of NIR
0.1	`publish`	`Unit`	Similar to standard publish with addition of NIR (1)
0.1	`nativeLink`	`File`	Link NIR and generate native binary
0.1	`nativeClang`	`File`	Path to `clang` command
0.1	`nativeClangPP`	`File`	Path to `clang++` command
0.1	`nativeCompileOptions`	`Seq[String]`	Extra options passed to clang verbatim during compilation
0.1	`nativeLinkingOptions`	`Seq[String]`	Extra options passed to clang verbatim during linking
0.1	`nativeMode`	`String`	Either `"debug"` or `"release"` (2)
0.2	`nativeGC`	`String`	Either `"none"` or `"boehm"` (3)

1. See Publishing and Cross compilation for details.
2. See Compilation modes for details.
3. See Garbage collectors for details.

Compilation modes

Scala Native supports two distinct linking modes:

1. **debug.**

v: latest

Default mode. Optimized for shortest compilation time. Runs fewer optimizations and is much more suited for iterative development workflow. Similar to clang's `-O0`.

2. **release.**

   Optimized for best runtime performance at expense of longer compilation time. Similar to clang's `-O2` with addition of link-time optimisation over the whole application code.

Garbage collectors

1. **boehm.**

   Conservative generational garbage collector. More information is available at the project's page.

1. **none.**

   Garbage collector that allocates things without ever freeing them. Useful for short-running command-line applications or applications where garbage collections pauses are not acceptable.

Publishing

Scala Native supports sbt's standard workflow for the package distribution:

1. Compile your code.
2. Generate a jar with all of the classfiles and NIR files.
3. Publish the jar to sonatype, bintray or any other 3rd party hosting service.

Once the jar has been published, it can be resolved through sbt's standard package resolution system.

Cross compilation

sbt-crossproject is an sbt plugin that lets you cross-compile your projects against all three major platforms in Scala: JVM, JavaScript via Scala.js and native via Scala Native. It's based on the original cross-project idea from Scala.js and supports the same syntax for exising JVM/JavaScript cross-projects. Please refer to project's README for details.

Continue to Language semantics.

v: latest