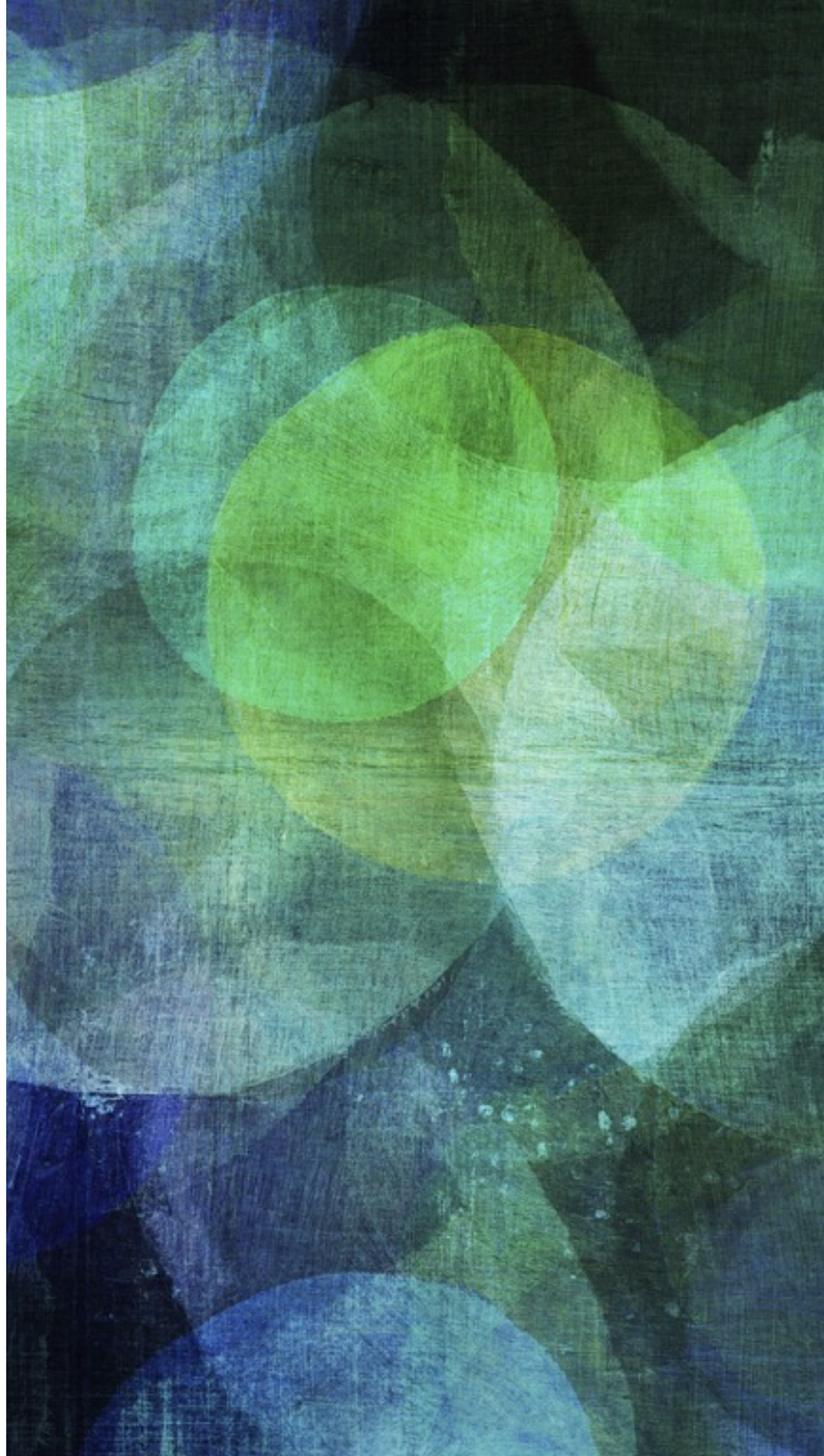
SCALA PART FIVE

SBT Basics



SBT SECTION OUTLINE

- What is it?
- Why?
- How?
  - Using IntelliJ
- Key concepts
- Examples
- External resources

SBT

- simple build tool
- scala build tool
- sane build tool

- there’ll be times you like it and other times

SBT WHAT IS IT?

- sbt is a build tool for Scala and Java. It requires Java 1.6 or later
- (ant, gradle, Make, etc)

SBT FEATURES

- Continuous compilation and testing with triggered execution
- Packages and publishes jars
- Supports mixed Scala/Java projects
- Starts the Scala REPL with project classes and dependencies on the classpath

SBT FEATURES

- Modularization supported with sub-projects
- External project support (e.g. list a git repository as a dependency)
- Parallel task execution, including parallel test execution
- Library management support: inline declarations, external Ivy or Maven configuration files, or manual management

SBT WHY USE IT?

- Keep your app dev organized, consistent amongst developers
- May be utilized in automated continuous integration build apps (jenkins, travis ci, team city, bamboo, etc.)

SBT HOW TO USE IT? PROJECT STRUCTURE CONVENTION

- project/ – project definition files
- project/*.scala – the main project definition file(s)
- project/build.properties – project, sbt and scala version definitions
- src/main – your app code goes here, in a subdirectory indicating the
  - code's language (e.g. src/main/scala, src/main/java)
- src/main/resources – static files you want added to your jar
  - (e.g. config files)
- src/test – like src/main, but for tests
- target – the destination for compilation (e.g. generate code, class files, jars)

SBT WITH INTELLIJ

- Let’s start with a simple as possible example
- HelloMundo

SBT CONSOLE EXAMPLES

- compile, run, package
- continuous compilation
- history and shortcuts
- scala console

SBT COMMON COMMANDS

Command	Description
clean	Removes all generated files from the target directory.
compile	Compiles source code files that are in src/main/scala, src/main/java, and the root directory of the project.
~ compile	Automatically recompiles source code files while you're running SBT in interactive mode (i.e., while you're at the SBT command prompt).
console	Compiles the source code files in the project, puts them on the classpath, and starts the Scala interpreter (REPL).
doc	Generates API documentation from your Scala source code using scaladoc.
help	Issued by itself, the help command lists the common commands that are currently available. When given a command, help provides a description of that command.
inspect	Displays information about. For instance, inspect library-dependencies displays information about the libraryDependencies setting. (Variables in *build.sbt* are written in camelCase, but at the SBT prompt, you type them using this hyphen format instead of camelCase.)

SBT COMMON COMMANDS CONTINUED . . .

Command	Description
package	Creates a JAR file (or WAR file for web projects) containing the files in src/main/scala, src/main/java, and resources in src/main/resources.
package-doc	Creates a JAR file containing API documentation generated from your Scala source code.
publish	Publishes your project to a remote repository. See Recipe 18.15, "Publishing Your Library".
publish-local	Publishes your project to a local Ivy repository. See Recipe 18.15, "Publishing Your Library". reload Reloads the build definition files (*build.sbt*, project/*.scala*, *and project*/.sbt), which is necessary if you change them while you're in an interactive SBT session.
run	Compiles your code, and runs the main class from your project, in the same JVM as SBT. If your project has multiple main methods (or objects that extend App), you'll be prompted to select one to run.
test	Compiles and runs all tests.
update	Updates external dependencies.

SBT HISTORY COMMANDS

- ! Show history command help.
- !! Execute the previous command again.
- !: Show all previous commands.
- !:n Show the last n commands.
- !n Execute the command with index n, as shown by the !: command.
- !-n Execute the nth command before this one.
- !string Execute the most recent command starting with 'string'
- !?string Execute the most recent command containing 'string'

SBT EXAMPLES

- External libraries dependencies
  - in build.sbt and project/build/*.scala
- Plugins
  - assembly

SBT EXTERNAL DEPENDENCIES IN BUILD.SBT

- Example build.sbt

```
name := "spark-cassandra-example"

version := "1.0"

assemblyOption in assembly := (assemblyOption in assembly).value.copy(includeScala = false)

// https://groups.google.com/a/lists.datastax.com/forum/#!topic/spark-connector-user/5muNwRaCJnU
assemblyMergeStrategy in assembly <<= (assemblyMergeStrategy in assembly) {
  (old) => {
    case PathList("META-INF", "io.netty.versions.properties") => MergeStrategy.last
    case x => old(x)
  }
}

scalaVersion := "2.10.6"

resolvers += "jitpack" at "https://jitpack.io"

libraryDependencies ++= Seq(
// use provided line when building assembly jar
// "org.apache.spark" %% "spark-sql" % "1.6.1" % "provided",
// comment above line and uncomment the following to run in sbt
   "org.apache.spark" %% "spark-sql" % "1.6.1",
   "com.datastax.spark" %% "spark-cassandra-connector" % "1.5.0"
)
```

SBT LIBRARY DEPENDENCIES

- http://www.scala-sbt.org/1.0/docs/Library-Dependencies.html

- Also noted in resources section

SBT CONCLUSION

- Any questions or comments?

- Resources:
  - Docs? http://www.scala-sbt.org/
  - Help? http://stackoverflow.com/questions/tagged/sbt