Chapter 13 : Building and Packaging

Exercises :

1. Create a new project in IntelliJ IDEA instead of importing one.
2. Install SBT in a Linux machine and use it.
3. Figure out how to change the configurations of SBT. For example, if you are behind a network proxy, how do you specify those settings in SBT, how do you change the heap size of SBT, etc.
4. Find out the meaning of “uber” or “Fat” JAR.
5. Figure out how to specify multiple dependencies in build.sbt.
6. Understand the different constructs of build.sbt and how to include different repositories.
7. Figure out how to specify multiple dependencies in build.sbt,
8. Figure out how to define multiple projects in build.sbt and how to structure your folders accordingly. Also, how do you selectively build/package individual projects in SBT.
9. Use maven to build Scala code instead of SBT. Understand the differences between the two.
10. Instead of specifying the required dependencies in build.sbt, find alternative ways to provide those dependencies for your project. (Hint: It involves putting JAR files in your project’s folder.)
11. Figure out how to launch the Scala shell and add different JARs to its classpa

Answers :

➔ uber-JAR

An **uber-JAR**—also known as a **fat JAR** or **JAR with dependencies**—is a JAR file that contains not only a Java program, but embeds its dependencies as well. This means that the JAR functions as an “all-in-one” distribution of the software, without needing any other Java code. (You still need a Java runtime, and an underlying operating system, of course.)

A single JAR file is simpler to deploy. There is no chance of mismatched versions of multiple JAR files. It is easier to construct a Java classpath, since only a single JAR needs to be included.

Disadvantages:

- Every time you need to update the version of the software, you must redeploy the entire uber-JAR. If you bundle individual JAR components, you need only update those that changed. This issue is of particular relevance to Java applications deployed via Java Web Start, since it automatically downloads the latest available version of each JAR

dependency; in that case, your application startup time will suffer if you use the uber-JAR.
- You cannot cherry-pick only the JARs containing the functionality you need, so your application's footprint may suffer from bloat.
- If downstream code relies on any of the same dependencies which are embedded in an unshaded uber-jar, you may run into trouble with multiple copies of those dependencies on your classpath, especially if you need to use a different version of that dependency than is bundled with the uber-JAR.

- SBT

SBT has been the most popular and the most used build tool in the Scala ecosystem for quite some time now. This wide acceptance and its variety of features make it the obvious choice for every Scala project. Another key point on why SBT is so popular is the fact that the build definition is written in Scala.

Similar to Maven, SBT can be downloaded as a compressed folder; the only installation needed is the uncompression. In addition to the unzip method, SBT can be installed via other tools, based on the platform you're on. For example, we can install SBT on a Mac with brew, with SDKMAN on a Debian-based Linux, and with the MSI installer on Windows.

The *build.sbt* file, similarly to the *pom.xml* for Maven projects, describes everything that happens during the build process.