**Q-Explain the working and the differences between Maven, Gradle and SBT in detail.**

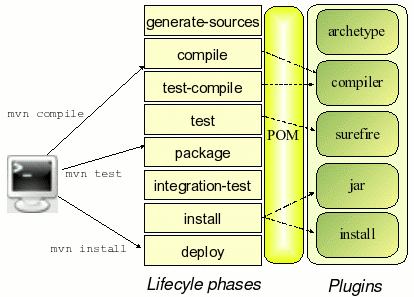
Answer:

**MAVEN:**

1. Maven is a "build management tool", it is for defining how your .java files get compiled to .class, packaged into .jar (or .war or .ear) files, (pre/post)processed with tools, managing your CLASSPATH, and all others sorts of tasks that are required to build your project.

2. It is similar to Apache Ant or Gradle or Makefiles in C/C++, but it attempts to be completely self-contained in it that you shouldn't need any additional tools or scripts by incorporating other common tasks like downloading & installing necessary libraries etc.

3. It is also designed around the "build portability" theme, so that you don't get issues as having the same code with the same buildscript working on one computer but not on another one (this is a known issue, we have VMs of Windows 98 machines since we couldn't get some of our Delphi applications compiling anywhere else).

4. Because of this, it is also the best way to work on a project between people who use different IDEs since IDE-generated Ant scripts are hard to import into other IDEs, but all IDEs nowadays understand and support Maven (IntelliJ, Eclipse) 

**SBT:**

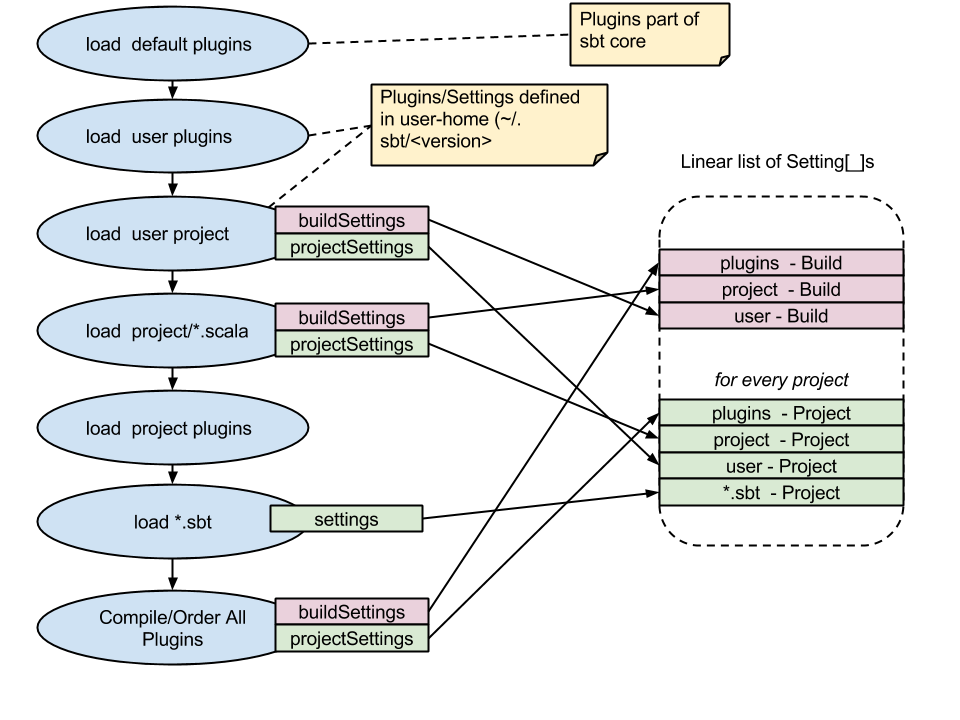
1. Sbt (simple build tool) is an open source build tool for Scala and Java projects, similar to Java’s Maven or Ant.

2. Two most prominent key features are incremental compilation and an interactive shell.

3. When continuous compilation mode is entered, the Scala compiler is instantiated only once which eliminates subsequent startup costs, and source file changes are tracked so that only affected dependencies are recompiled.

4. The interactive console allows modifying build settings on the fly and entering the Scala REPL (Read-Evaluate-Print Loop) along with all class files of the project.

5. The popularity of the incremental compilation has prompted Typesafe to extract this feature in the form of an independent component called Zinc.



**GRADLE:**

*1. Gradle* is an advanced general purpose build management system based on Groovy and Kotlin.

*2.* Gradle supports the automatic download and configuration of dependencies or other libraries. It supports Maven and Ivy repositories for retrieving these dependencies. This allows reusing the artifacts of existing build systems.

3. Gradle supports multi-project and multi-artifact builds.

4. Gradle has the notion of *projects* and *tasks*.

5. A Gradle build consists of one or more *projects*.

6. Projects can be something which should be built or something that should be done.

7. Each project consists of *tasks*. A task represents a piece of work which a build performs, e.g., compile the source code or generate the Javadoc.

8. A project using Gradle describes its build via a *build.gradle* file. This file is located in the root folder of the project.

9. The build file for Gradle builds is based on a *Domain Specific Language* (DSL). In this file you can use a combination of declarative and imperative statements. You can also write Groovy or Kotlin code, whenever you need it.

10. Tasks can also be created and extended dynamically at runtime.

11. This build file defines a project and its tasks. Gradle is a general purpose build system hence this build file can perform any task.

