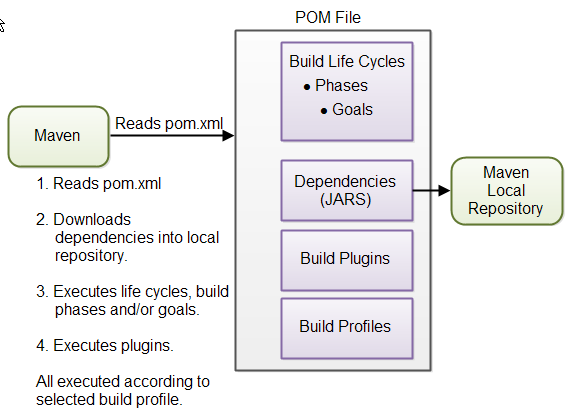
Assignment-15.2:

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

Maven:

Maven is a "build management tool", it is for defining how your .java files get compiled to .class, packaged into .jar, managing your classpath, and all others sorts of tasks that are required to build your project.  It is similar to Apache Ant or Gradle. It does everything on its own we don’t need any additional tools or scripts incorporating other common tasks like downloading & installing necessary libraries etc. 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. 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



Gradle:

Gradle is an advanced general purpose build management system based on Groovy and Kotlin. 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. Gradle supports multi-project and multi-artifact builds.

* **Declarative build**- Gradle provides declarative language elements that we can be assembled as per our choice. This declarative language is extensible which enables us to add our own new language or enhance the existing one. These elements also provide build by convention support for Java, Groovy and other modern Java based technologies.
* **Language for dependency based programming**- The declarative language is very flexible and enables Gradle to support the specific requirements.
* **Structure the build**- Gradle allows us to apply common design principles to our build, which enables us to create a structured build for our applications.
* **Scalability**- Gradle has the ability to easily scale from a simple single project build to a huge enterprise multi-project build. It enables the incremental build and also has the ability to tackle the performance issues that plague large enterprise build scripts.
* **Multi-project support**- Gradle supports multi-project builds. It enables us to maintain the relationships between different projects in the case of a multi-project build environment. It also supports partial builds. We can build a single subproject out of an enterprise application. While building the single subproject, Gradle takes care of the other subprojects if the said subproject has dependency on other subprojects.

**SBT**:

**SBT** is an [open Source](https://en.wikipedia.org/wiki/Open_source) [Build Tool](https://en.wikipedia.org/wiki/Build_tool) for [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)) and [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) projects, similar to Java's [Maven](https://en.wikipedia.org/wiki/Apache_Maven) or [Ant](https://en.wikipedia.org/wiki/Apache_Ant).

The functionality of sbt can be extended through a plugin architecture. A dedicated website was set up for community contributed plugins, which cover various areas such as signing, packaging, publishing and releasing artifacts, connecting to other services such as blogs and databases, or integrating with other technologies such as deploying to the Android platform.

There are plugins to automatically create project files for the Eclipse and IntelliJ IDEA IDEs. On the other hand, an IntelliJ IDEA plugin allows the sbt console to be integrated into IDEA, and projects can choose to use sbt for building.

Its main features are:

* native support for compiling Scala code and integrating with many Scala [test frameworks](https://en.wikipedia.org/wiki/Test_automation)
* build descriptions written in Scala using a [DSL](https://en.wikipedia.org/wiki/Domain_Specific_Language)
* dependency management using [Ivy](https://en.wikipedia.org/wiki/Apache_Ivy) (which supports Maven-format repositories)
* continuous compilation, testing, and deployment
* integration with the Scala interpreter for rapid iteration and debugging
* support for mixed Java/Scala projects