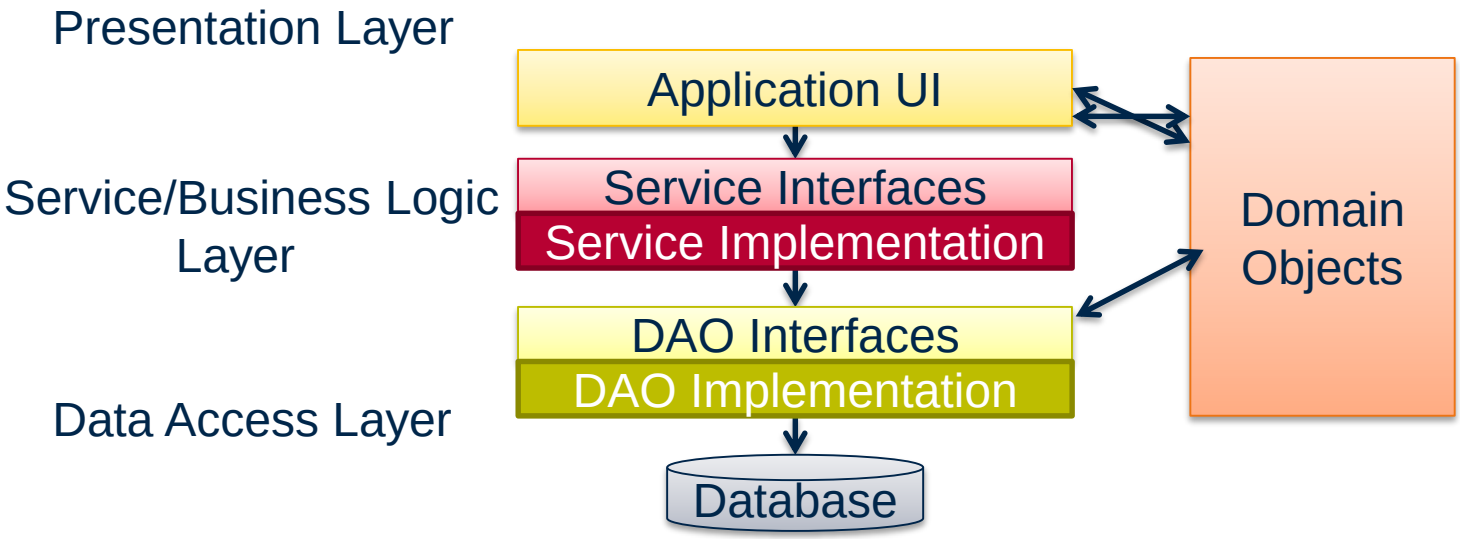
Introduction to Layered Architecture
&
Maven build Tool

Lesson Objectives

- After completing this lesson, participants will be able to
  - Understand the concept of Layered Architecture
  - Implement layers in Java applications

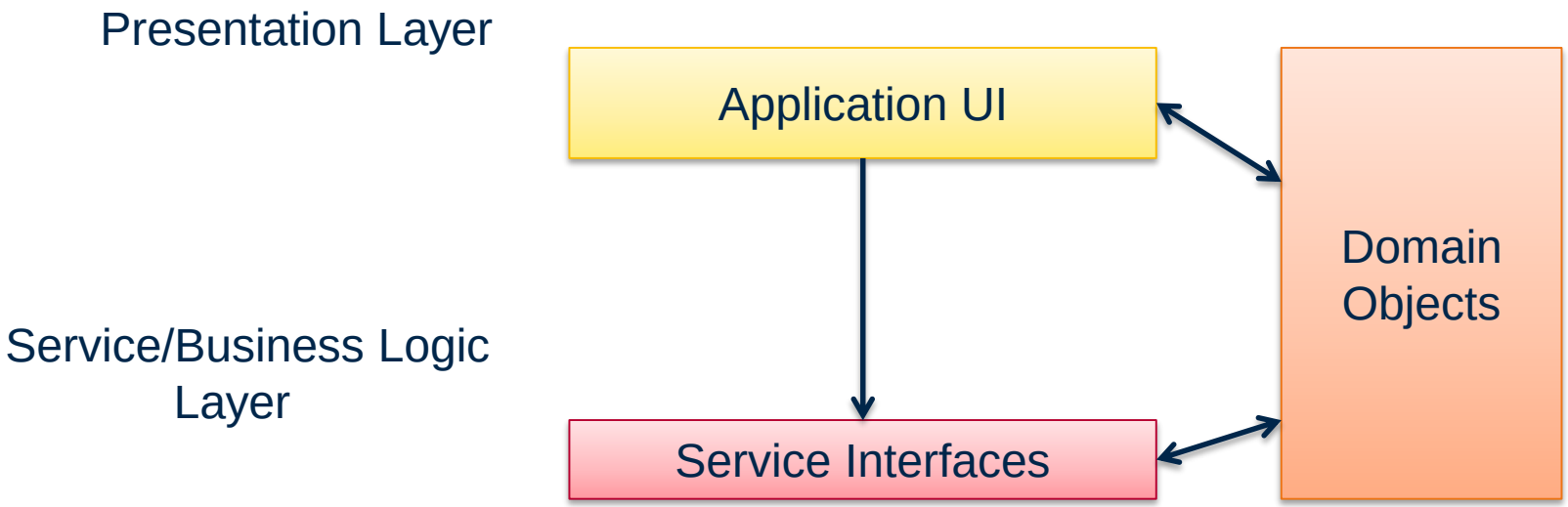
What is Layered Architecture?

- Layered architecture is one of the architectural pattern based on call-and-return style
- In layered architecture, business rules, behavior, and data are obtained and manipulated, based on activity via the user interface.
- Layered architecture provides a clean separation between the business implementation, presentation and data-access logic.



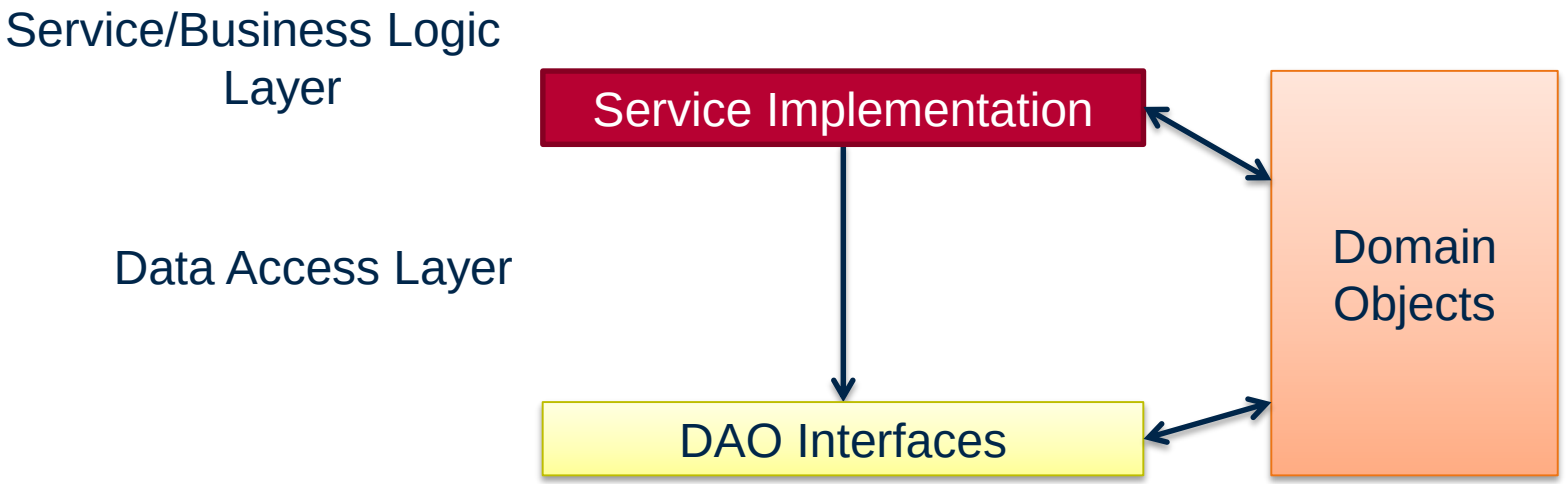
Presentation Layer

- Presentation layer consists of objects defined to accept user input and to display application outputs
- Exception handling is also an important responsibility of this layer.
- Presentation-layer simply request service/business layer for required functionality by sending and receiving domain objects



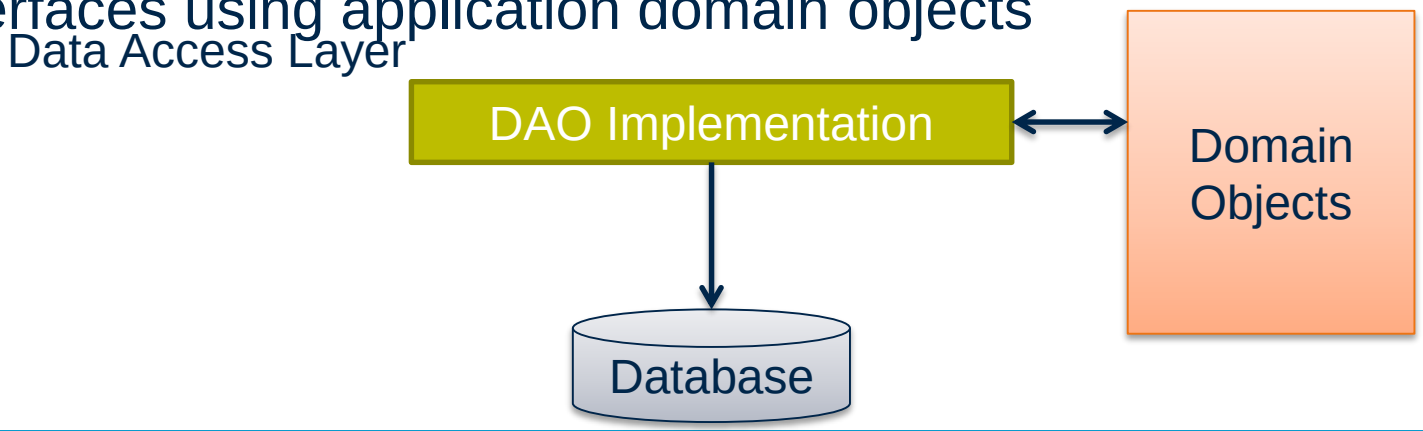
Business Logic/Service Layer

- Business logic layer is concerned with the retrieval, processing, transformation and management of application data
- This layer is responsible to implement business rules and policies
- It also ensures data consistency and validity
- Presentation layer passes data collected from UI to business layer and interact with business logic through abstract interfaces



Data Access Layer

- This layer abstract the logic required to access the underlying data stores
- It centralize common data access functionality in order to make the application easier to configure and maintain.
- This layer is responsible for managing connections, generating queries, and mapping application domain objects to data source structures
- Business logic layer interacts to data access layer through abstract interfaces using application domain objects

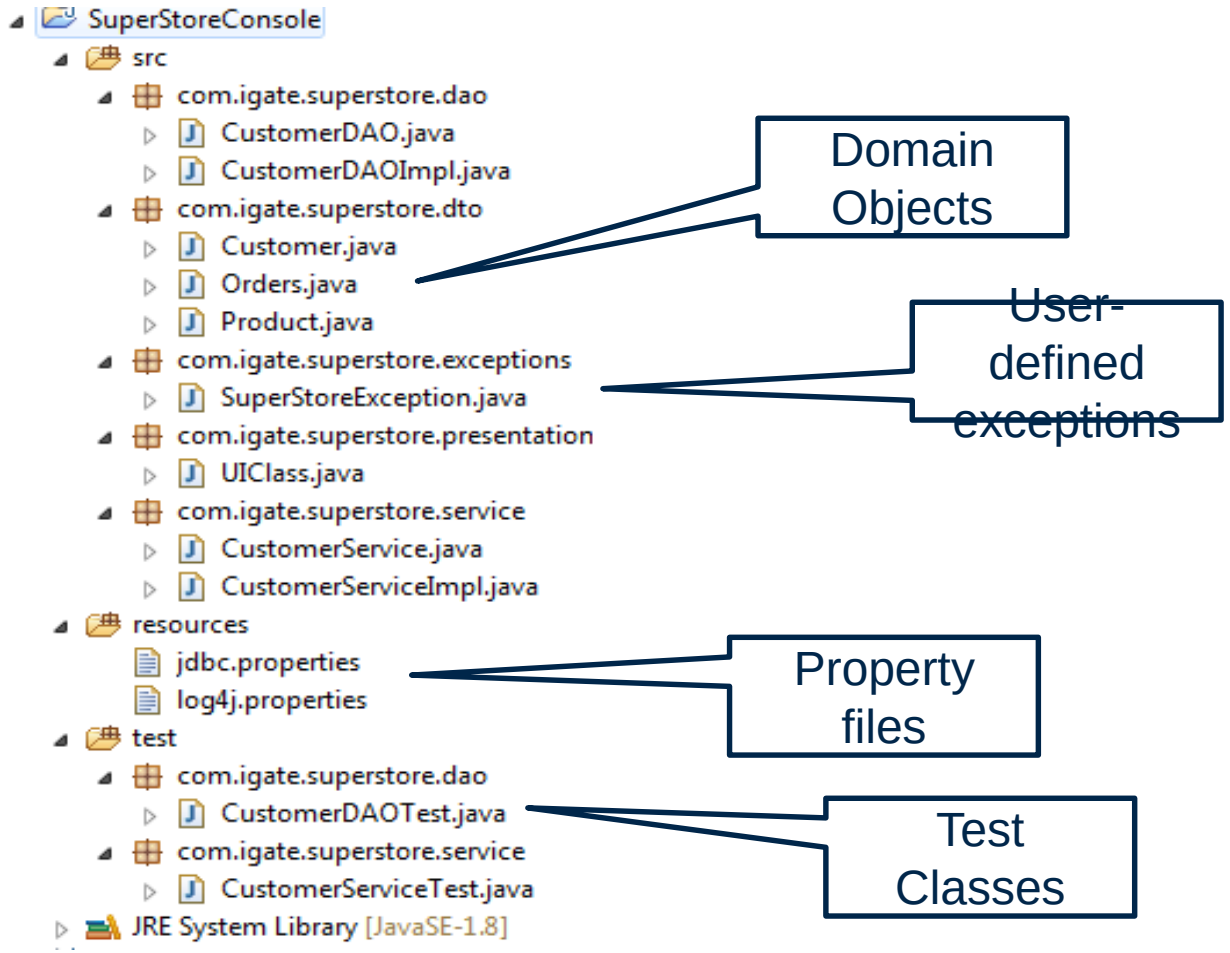


Data Transfer Objects

- Data transfer objects (DTO) or Value Objects (VO) encapsulates business data necessary to represent real world elements, such as Customers or Orders
- These object are POJO's to store data values and expose them through properties
- They contain and manage business data used by the entire application

Data Transfer Objects

Sample Layered Application Structure



LayeredApp.zip

Introduction to Apache Maven | A build automation tool for Java projects

Apache Maven | A build automation tool for Java projects

Maven is a powerful project management tool that is based on POM (project object model).

It is used for projects build, dependency and documentation. It simplifies the build process like ANT. But it is too much advanced than ANT.

In short terms we can tell maven is a tool that can be used for building and managing any Java-based project.

Maven makes the day-to-day work of Java developers easier and generally help with the comprehension of any Java-based project.

A build automation tool for Java projects

Maven does a lot of helpful task like

1.We can easily build a project using maven.

2.We can add jars and other dependencies of the project easily using the help of maven.

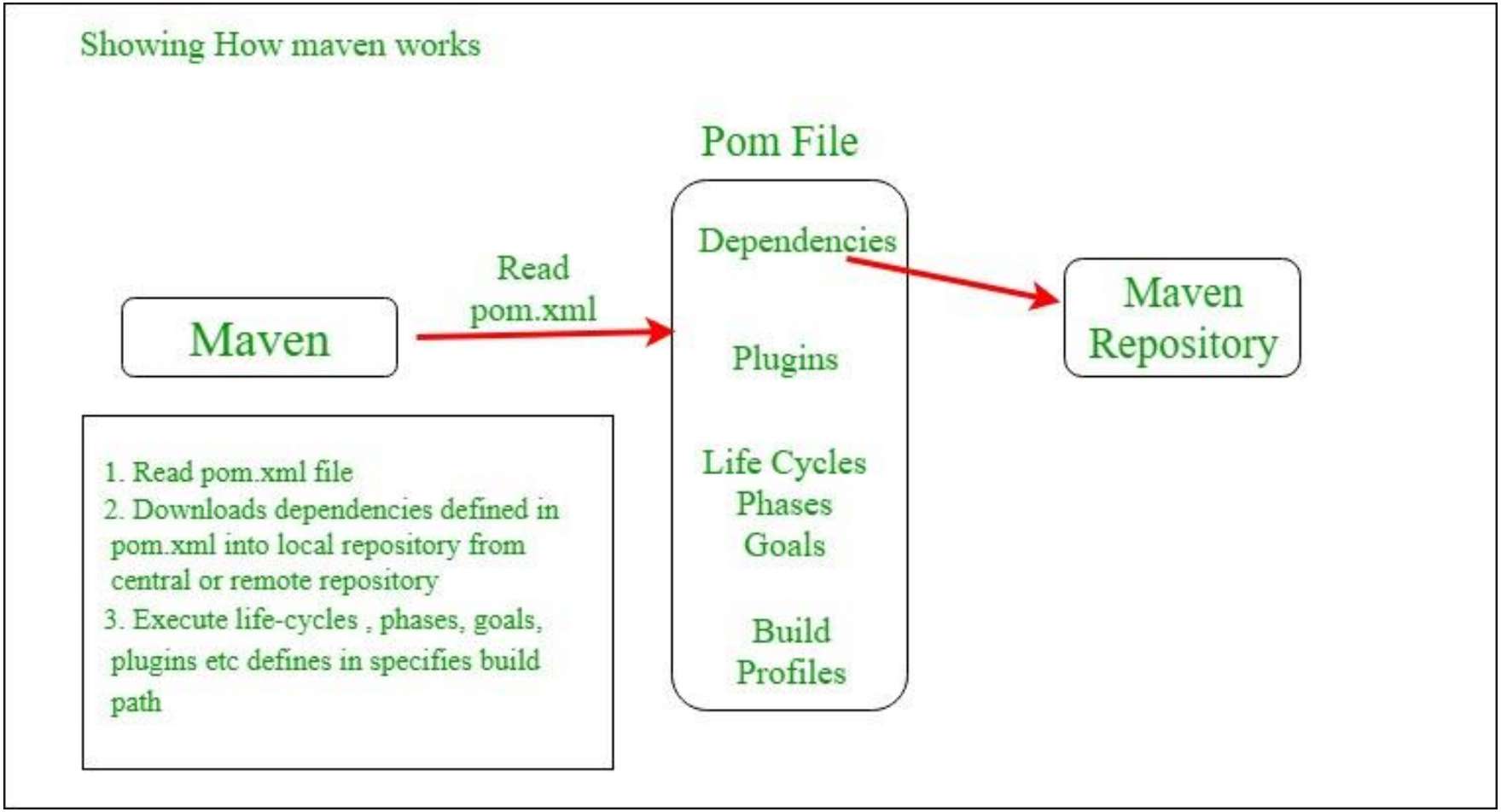
3.Maven provides project information (log document, dependency list, unit test reports etc.)

4.Maven is very helpful for a project while updating central repository of JARs and other dependencies.

5.With the help of Maven we can build any number of projects into output types like the JAR, WAR etc without doing any scripting.

6.Using maven we can easily integrate our project with source control system (such as Subversion or Git).

How maven works?



Core Concepts of Maven

1.POM Files: Project Object Model(POM) Files are XML file that contains information related to the project and configuration information such as dependencies, source directory, plugin, goals etc. used by Maven to build the project

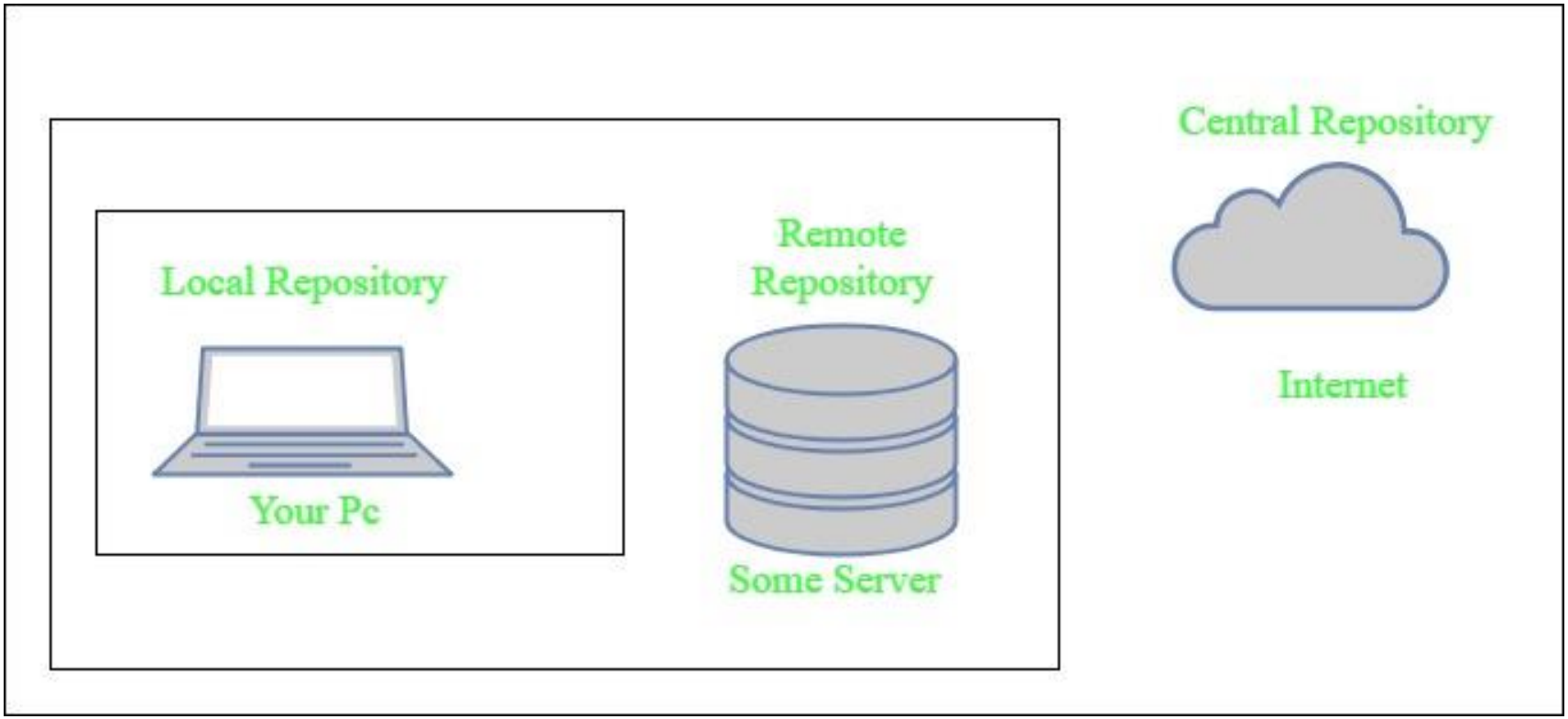
2.Dependencies and Repositories: Dependencies are external Java libraries required for Project and repositories are directories of packaged JAR files.

3.Build Life Cycles, Phases and Goals: A build life cycle consists of a sequence of build phases, and each build phase consists of a sequence of goals. Maven command is the name of a build lifecycle, phase or goal.

4.Build Profiles: Build profiles a set of configuration values which allows you to build your project using different configurations. For example, you may need to build your project for your local computer, for development and test. To enable different builds you can add different build profiles to your POM files.

5.Build Plugins: Build plugins are used to perform specific goal. you can add a plugin to the POM file. Maven has some standard plugins you can use, and you can also implement your own in Java.

Repositories



Sample pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

                <modelVersion>4.0.0</modelVersion>
                <groupId> com.project.loggerapi </groupId>
                <artifactId>LoggerApi</artifactId>
                <version>0.0.1-SNAPSHOT</version>

        <!-- Add typical dependencies for a web application -->
        <dependencies>
                <dependency>
                        <groupId>org.apache.logging.log4j</groupId>
                        <artifactId>log4j-api</artifactId>
                        <version>2.11.0</version>
                </dependency>
        </dependencies>

</project>
```

Sample pom.xml

Elements used for Creating pom.xml file

1.project- It is the root element of the pom.xml file.

2.modelVersion- modelversion means what version of the POM model you are using. Use version 4.0.0 for maven 2 and maven 3.

3.groupId- groupId means the id for the project group. It is unique and Most often you will use a group ID which is similar to the root Java package name of the project like we used the groupId com.project.loggerapi.

4.artifactId- artifactId used to give name of the project you are building.in our example name of our project is LoggerApi.

5.version- version element contains the version number of the project. If your project has been released in different versions then it is useful to give version of your project.

Sample pom.xml

Other Elements of Pom.xml file

1.dependencies- dependencies element is used to defines a list of dependency of project.

2.dependency- dependency defines a dependency and used inside dependencies tag. Each dependency is described by its groupId, artifactId and version.

3.name- this element is used to give name to our maven project.

4.scope- this element used to define scope for this maven project that can be compile, runtime, test, provided system etc.

5.packaging- packaging element is used to packaging our project to output types like JAR, WAR etc.

Thank You!