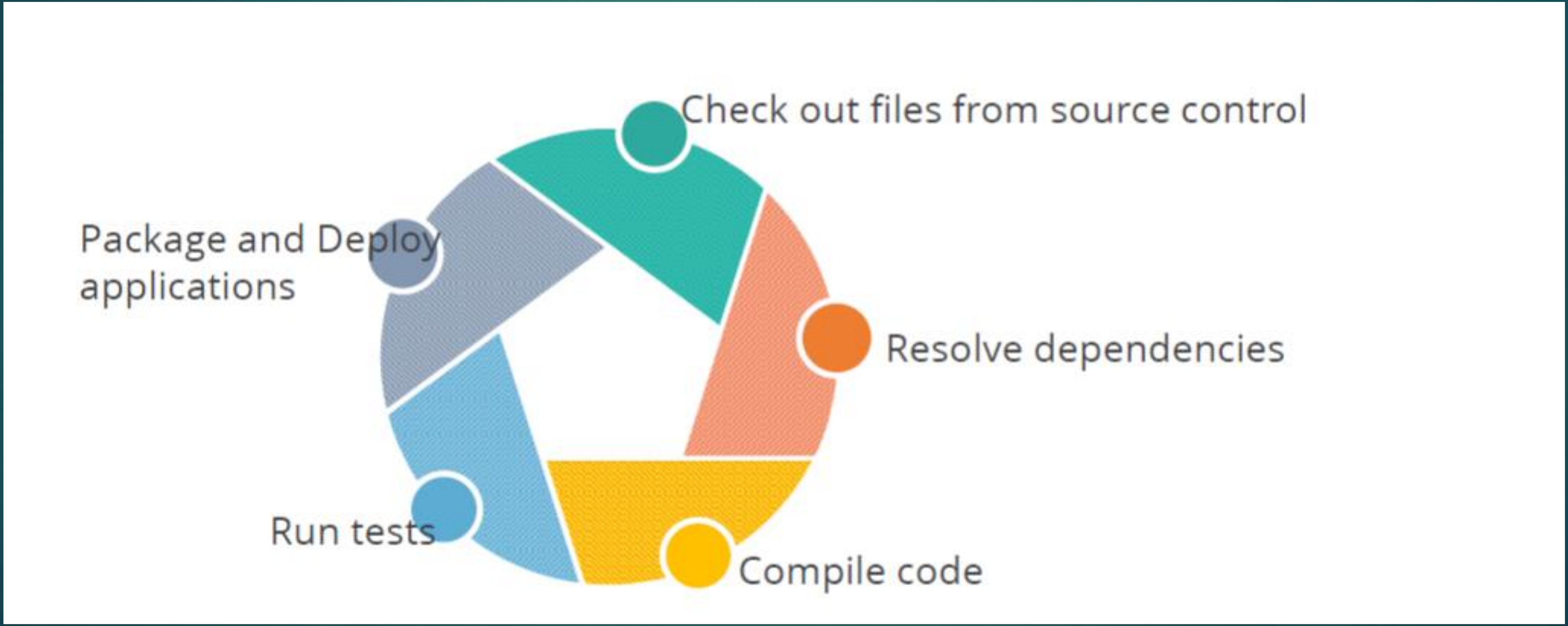
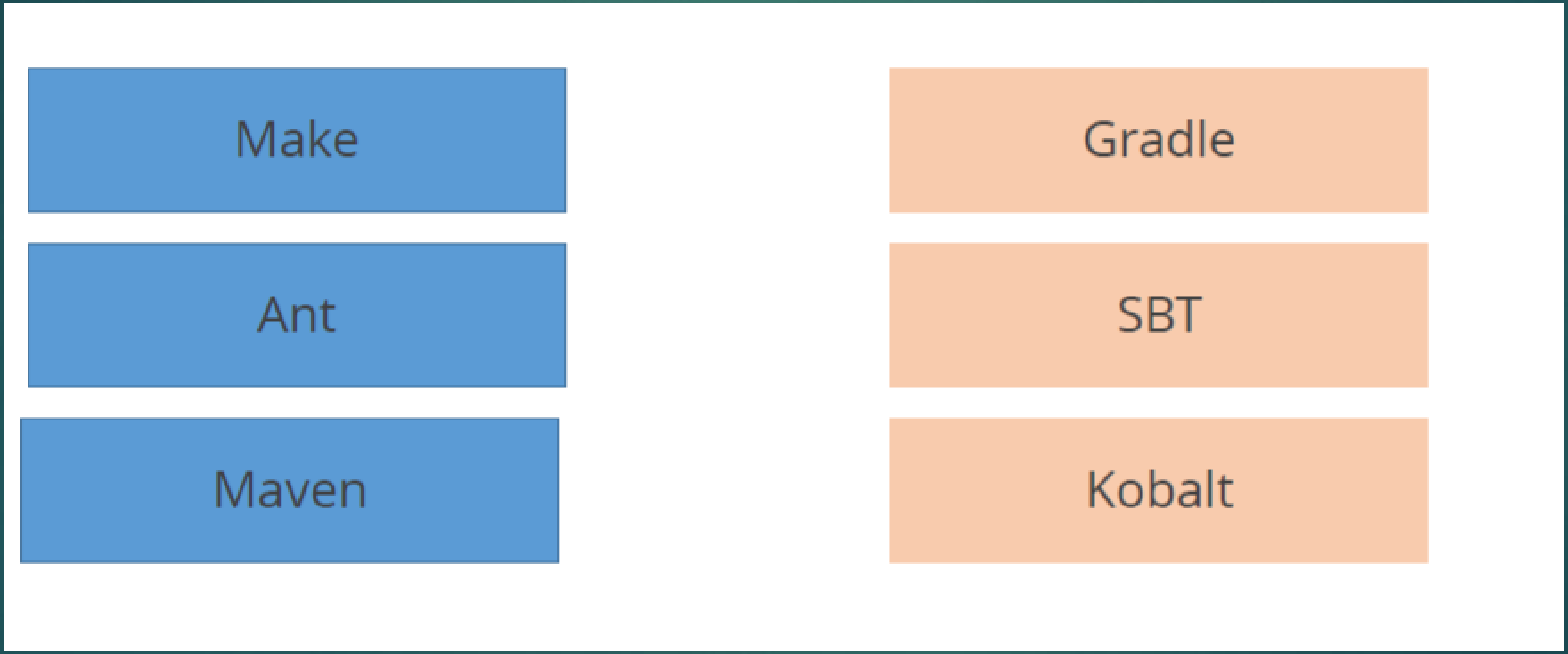
Build Tools

Build Tools are used to manage the software lifecycle


Check out files from source control
Resolve dependencies
Compile code
Run tests
Package and Deploy applications

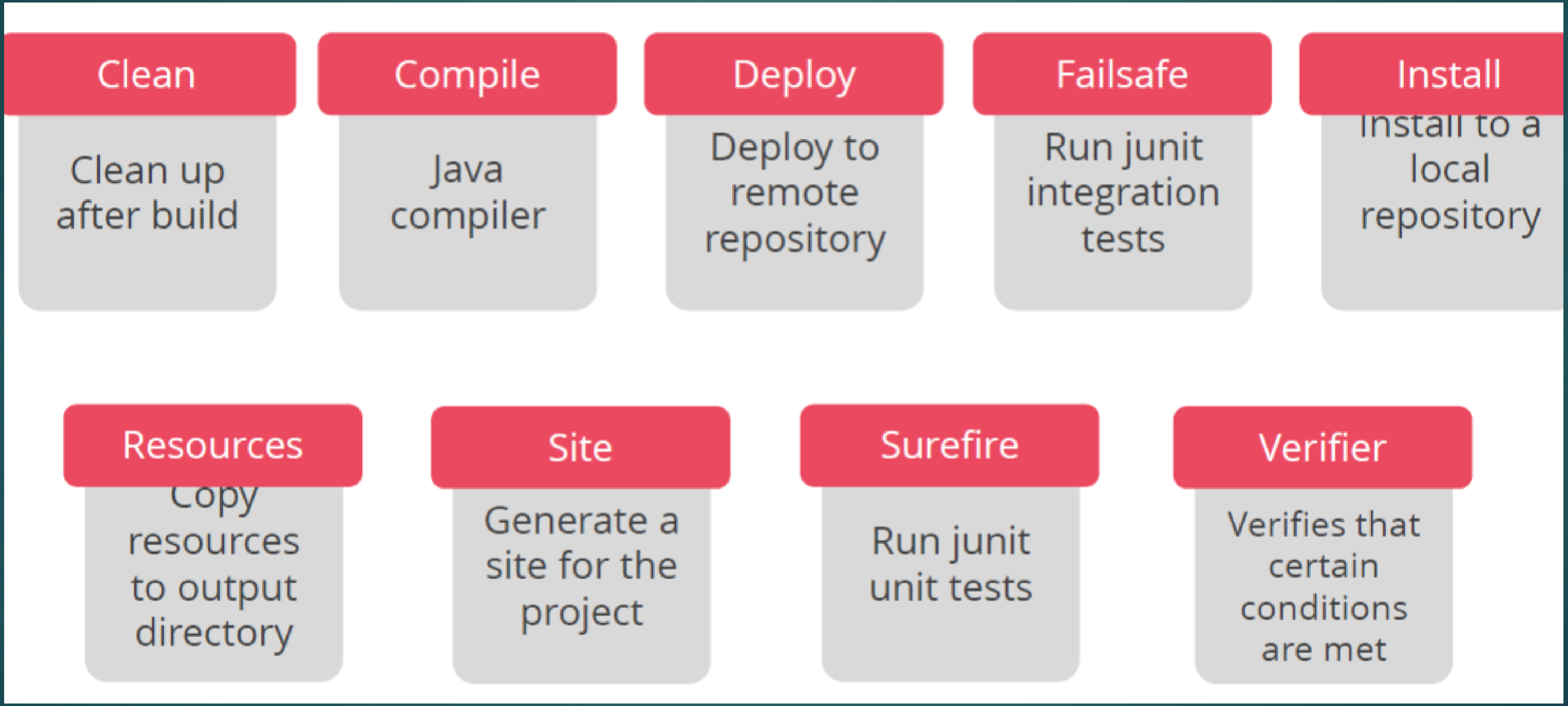
Various Build Tools



Maven

- Maven was created to allow a developer to understand the state of a project quickly
- Making the build process easy
- Providing a uniform build system
- Providing quality project information
- Providing guidelines for best practices development
- Allowing transparent migration to new features

Maven – Core Plugins



Clean
Clean up after build

Compile
Java compiler

Deploy
Deploy to remote repository

Failsafe
Run junit integration tests

Install
Install to a local repository

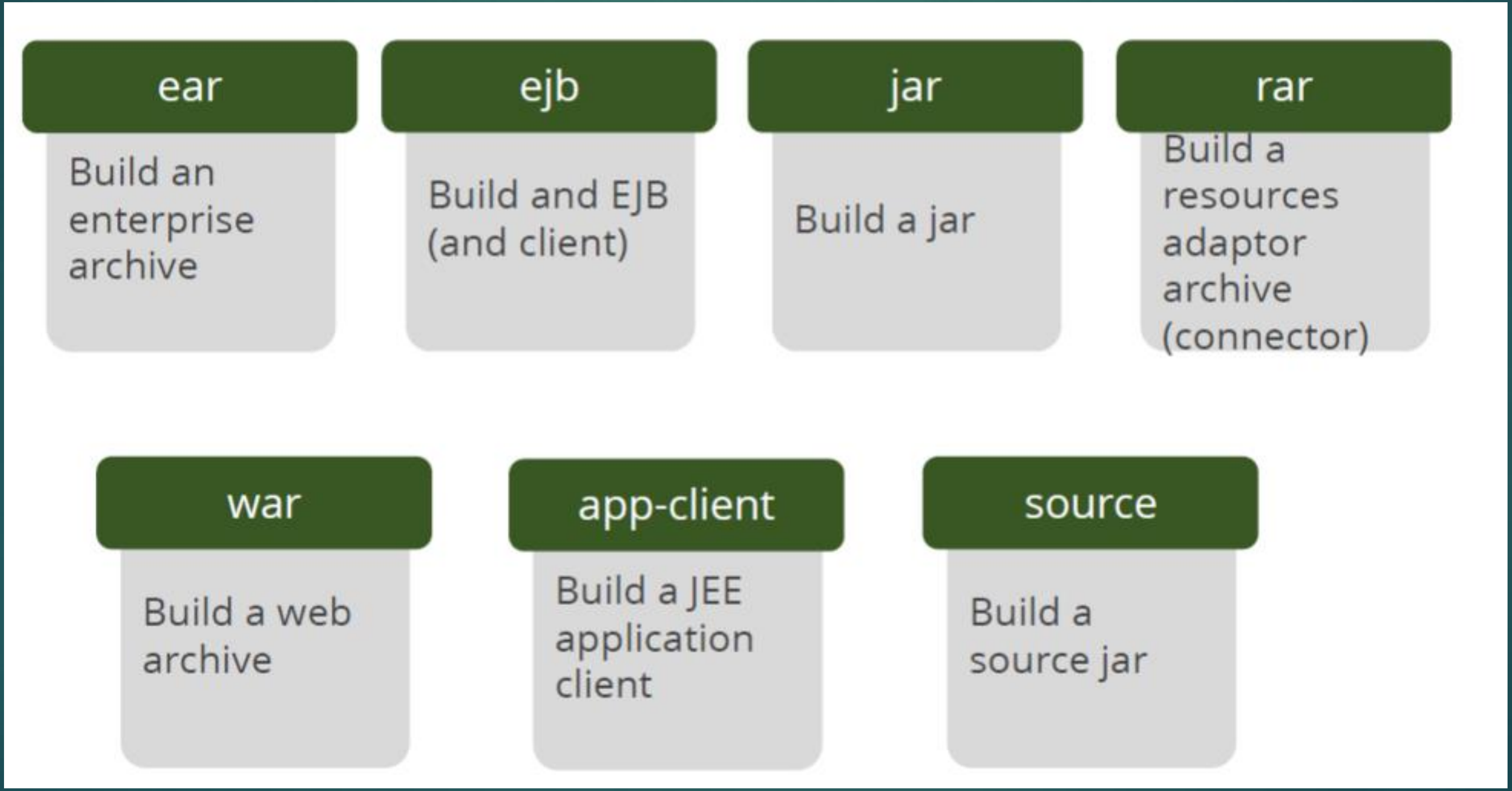
Resources
Copy resources to output directory

Site
Generate a site for the project

Surefire
Run junit unit tests

Verifier
Verifies that certain conditions are met

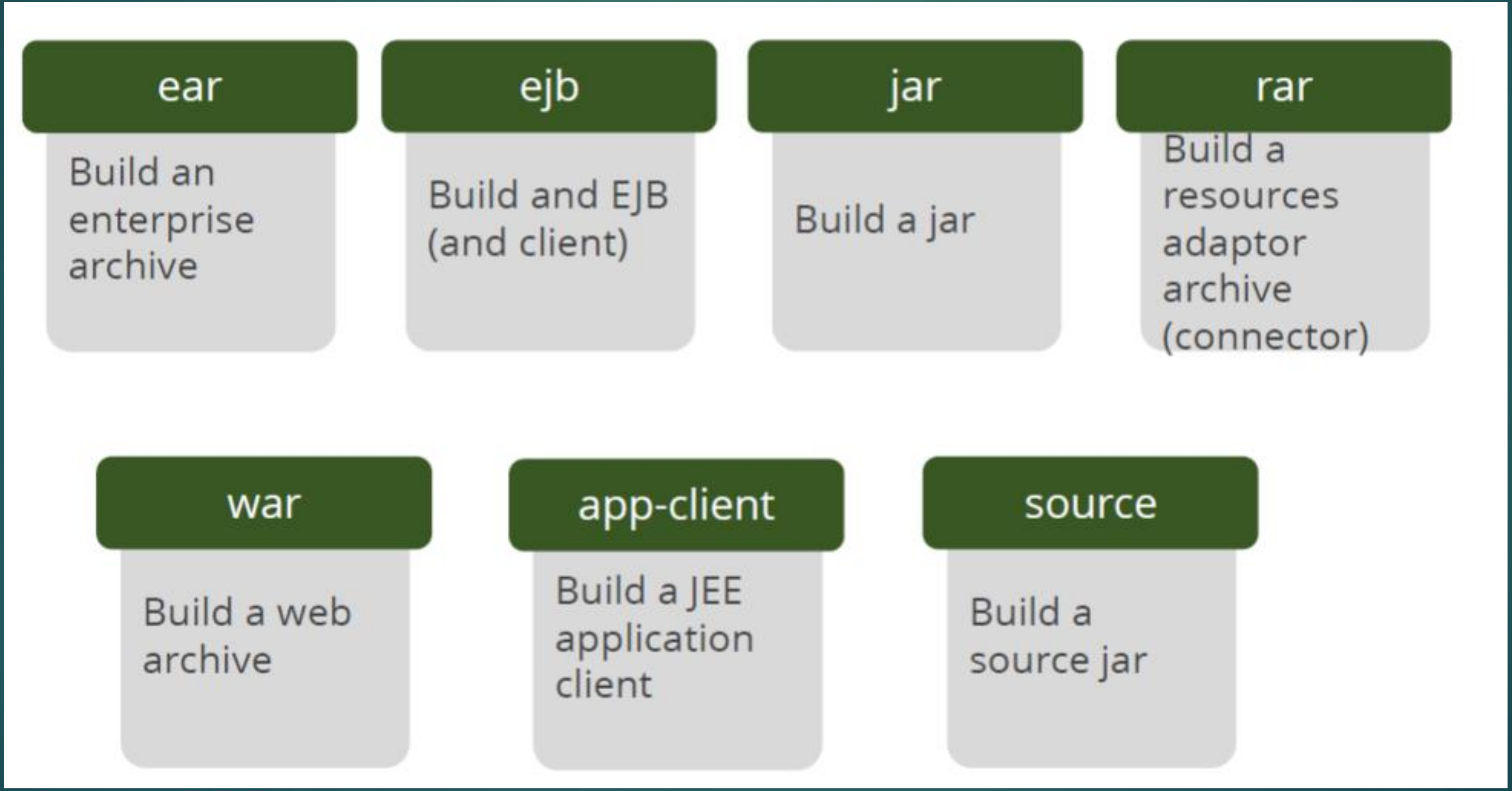
Maven – Packaging Plugins



ear	ejb	jar	rar
Build an enterprise archive	Build and EJB (and client)	Build a jar	Build a resources adaptor archive (connector)

war	app-client	source
Build a web archive	Build a JEE application client	Build a source jar

Maven Reporting Plugins


ear
Build an enterprise archive
ejb
Build and EJB (and client)
jar
Build a jar
rar
Build a resources adaptor archive (connector)
war
Build a web archive
app-client
Build a JEE application client
source
Build a source jar

Maven Archetypes

- Archetype is a Maven project templating toolkit
- Projects can be created from archetypes
- An archetype is selected from a catalogue
- The archetype is configured
- The project is created

mvn archtetype:generate

Project Object Model

Project Object Model (POM)

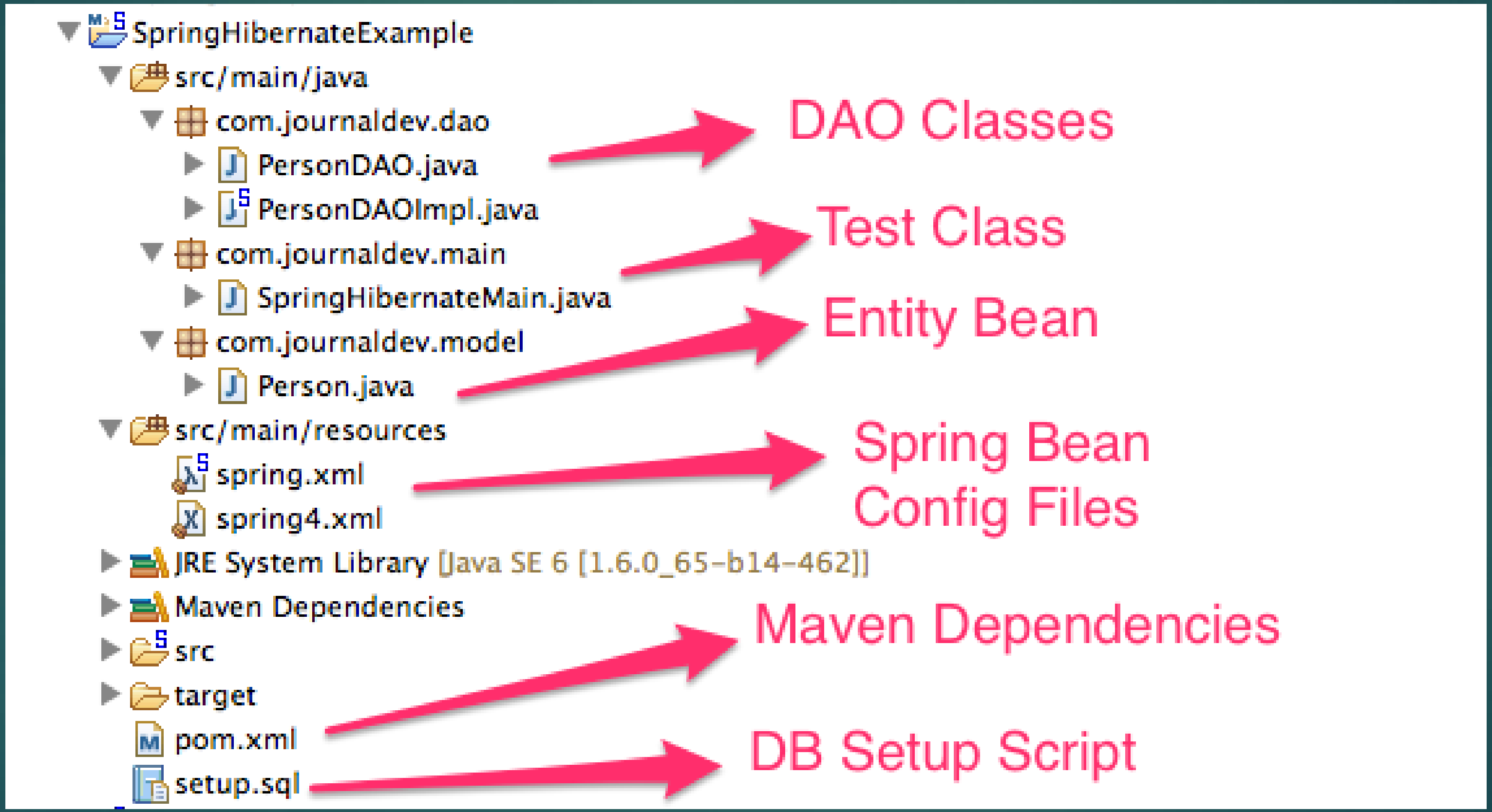
- An XML representation of a Maven project
- In the file pom.xml at the root of a project

A POM is declarative

- Declares who, what, and where

- A Maven project can simply be the pom.xml file
- Most projects have code associated with them

Maven in Spring Framework



```
▼ SpringHibernateExample
  ▼ src/main/java
    ▼ com.journaldev.dao
      ▶ PersonDAO.java
      ▶ PersonDAOImpl.java
    ▼ com.journaldev.main
      ▶ SpringHibernateMain.java
    ▼ com.journaldev.model
      ▶ Person.java
  ▼ src/main/resources
      spring.xml
      spring4.xml
  ▶ JRE System Library [Java SE 6 [1.6.0_65-b14-462]]
  ▶ Maven Dependencies
  ▶ src
  ▶ target
    pom.xml
    setup.sql
```

POM Basics and Dependencies

- The POM needs to define the Group ID, Artifact ID, and Version
- The packaging should also be declared – the default is jar
- Dependencies can be added to the POM file
- Dependencies are downloaded from maven repositories and added to the build

https://hybrisdiary.com/2017/07/05/spring-boot/

Gradle Features

- **Declarative builds and build-by-convention** – Gradle is available with separate Domain Specific Language (DSL) based on Groovy language. Gradle provides declarative language elements. The elements also provide build-by-convention support for Java, Groovy, OSGi, Web and Scala.
- **Language for dependency based programming** – The declarative language lies on top of a general purpose task graph, which you can fully leverage in your build.
- **Structure your build** – Gradle allows you to apply common design principles to your build. It gives you a perfect structure for build, so that you can design well-structured and easily maintained, comprehensible build.
- **Deep API** – Using this API, you can monitor and customize its configuration and execution behavior to its core.
- **Gradle scales** – Gradle can easily increase productivity, from simple and single project builds to huge enterprise multi-project builds.
- **Multi-project builds** – Gradle supports multi-project builds and also partial builds. If you build a subproject, Gradle takes care of building all the subprojects that it depends on.

Gradle Features

- **First build integration tool** – Gradle completely supports ANT tasks, Maven and Ivy repository infrastructure for publishing and retrieving dependencies. It also provides a converter for turning a Maven pom.xml to Gradle script.
- **Ease of migration** – Gradle can easily adapt to any structure you have. Therefore, you can always develop your Gradle build in the same branch where you can build
- **Gradle Wrapper** – Gradle Wrapper allows you to execute Gradle builds on machines where Gradle is not installed. This is useful for continuous integration of servers.
- **Free open source** – Gradle is an open source project, and licensed under the Apache Software License (ASL).
- **Groovy** – Gradle's build script is written in Groovy. The whole design of Gradle is oriented towards being used as a language, not as a rigid framework. Groovy allows you to write your own script with some abstractions. The entire Gradle API is designed in Groovy language.

Gradle vs Maven

https://gradle.org/gradle-vs-maven-performance/

Building Java projects with Gradle

 https://spring.io/guides/gs/gradle/

Building Java projects with Maven

https://spring.io/guides/gs/maven/

THANK YOU