DATE : 09.05.2025

DT/NT : NT

LESSON : DEVOPS

SUBJECT: MAVEN

BATCH : B 303

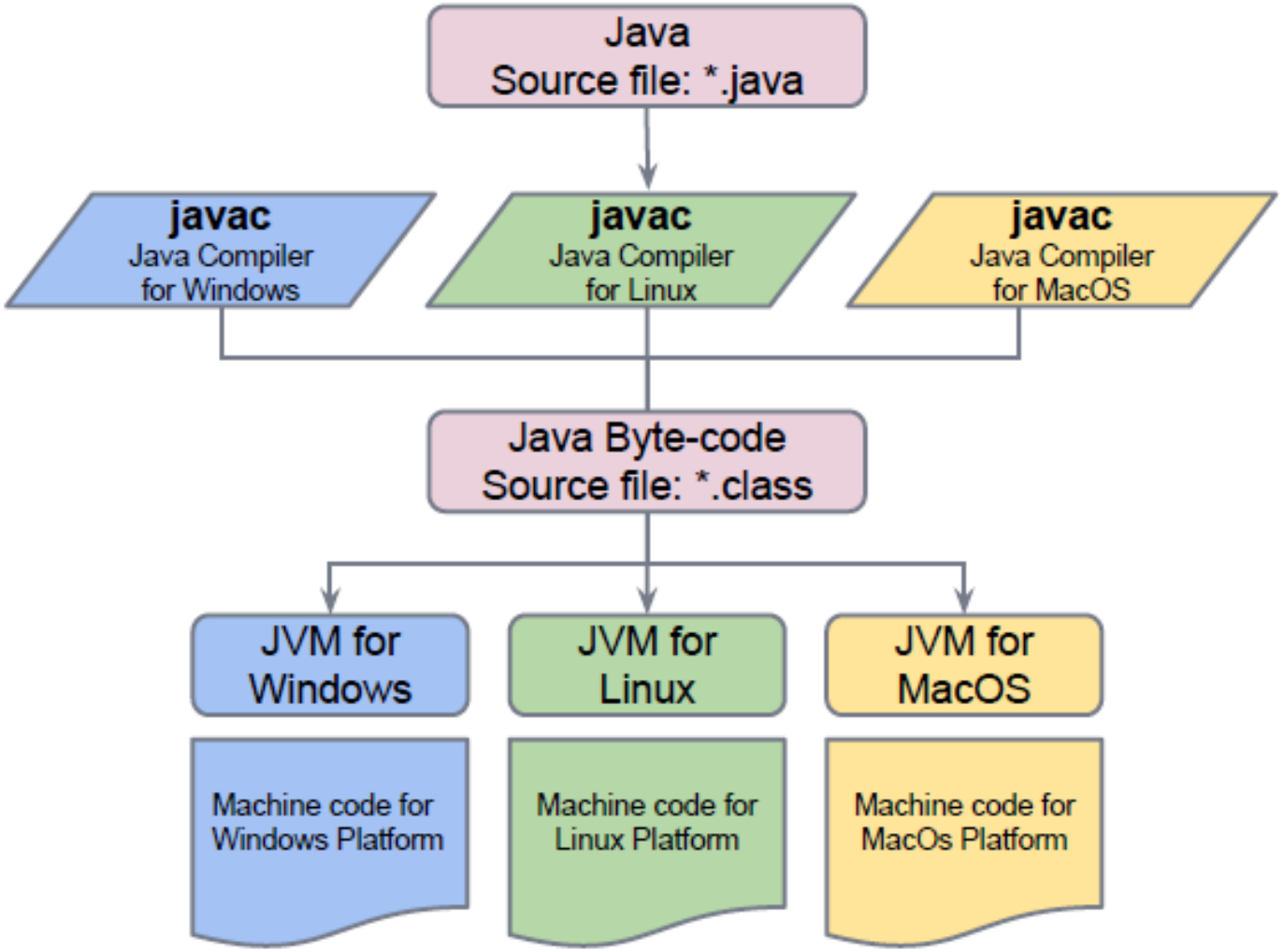
AWS-DEVOPS

TECHPRO EDUCATION

Introduction to Java

- Java is a **general-purpose programming language**
- That is **class-based, object-oriented**, and designed to have **as few dependencies as possible**
- It is intended to **Write Once, Run Anywhere (WORA)**
- Applications are **compiled** to **bytecode** that can run on any **Java Virtual Machine (JVM)**
- **Sun Microsystems** released the first public implementation as **Java 1.0 in 1996**
- Java software runs on everything **from laptops to data centers, game consoles to scientific supercomputers.**

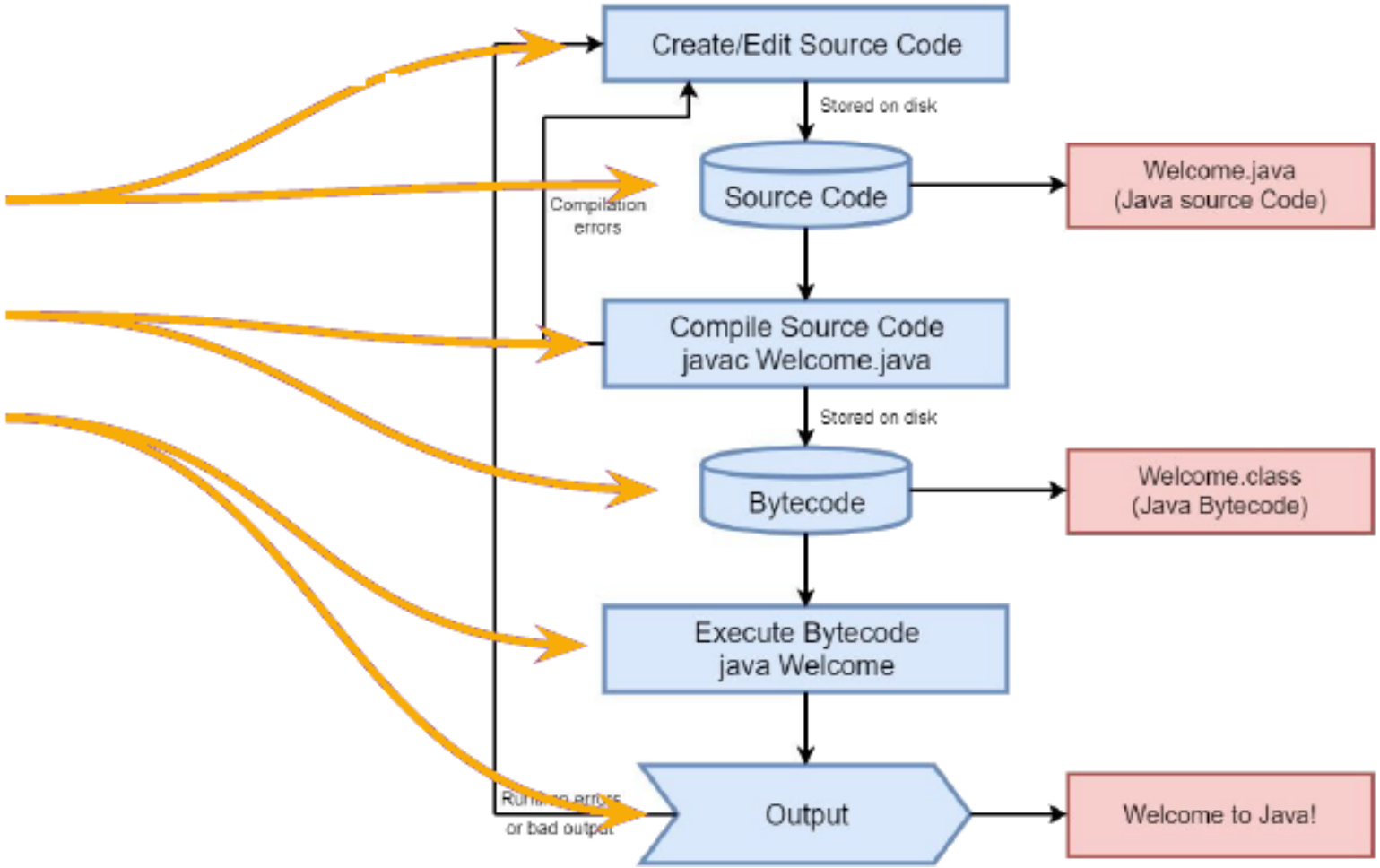
Java Specification



Java Specification

Create, Compile and Run

- Create
- Compile
- Run



What is Maven

Table of Contents

Introduction to Maven

- First, it was used at **Apache's Jakarta Alexandria Project** in 2001
- What Maven did was **to simplify the build processes**

Introduction to Maven

As a project management tool, Maven :

- builds **multiple projects** easily,
- **publishes documentation** for the projects,
- accomplishes an **easy deployment**,
- **helps in collaboration** with development teams.

Introduction to Maven

Maven can :

- **manage the versions** of consecutive builds,
- **compile** source code into binary,
- **download dependencies**,
- **run tests**,
- **package** compiled code
- **deploy** artifacts

Features of Maven

- **Easy to start** with Maven
- Variety of **options**
- **Same structure** across different projects
- **Easy to integrate** into a developing team
- It has a **powerful dependency management tool**
- **Large repository** of libraries

Features of Maven

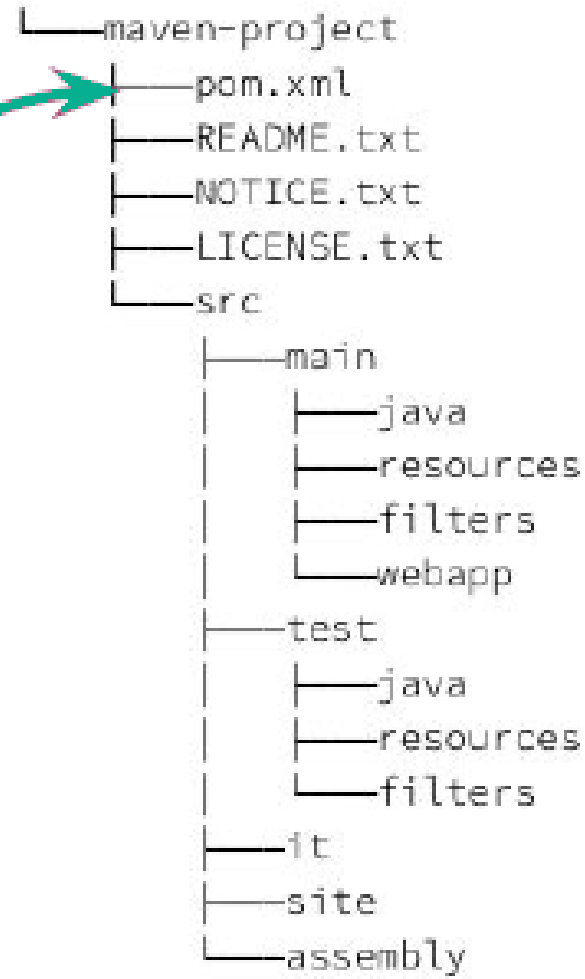
- **Extra features** with plugins
- **Different outputs** like a **jar**, **ear** or **war**
- Maven can **generate a website**
- Maven can **support the older versions**

Directory Structure

Project structure **should conform** to →

The most important file is the **pom file** →

- defines project's **config details**



POM File

Introduction to POM File

- It is an XML file
- **P**roject **O**bject **M**odel is the **starting point** for a Maven project
- It contains **configurations** about the project
- When a task or goal is executed, **Maven searches for** the **POM file**

Introduction to POM File

POM defines

- Project **dependencies**
- Plugins and **goals** to be executed
- **Build profiles**
- Other information like the **project version**, **description**,**developers**, **mailing lists**, and more...

Introduction to POM File

- There **must** be a POM file in every Maven project
- **All POMs need** at least
  - Project tag
  - modelVersion tag
  - **g**roupId tag
  - **a**rtifactId tag
  - **v**ersion (Last three called as **gav** in short)

```
1  <project xmlns = "http://maven.apache.org/POM/4.0.0"
2  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
3  xsi:schemaLocation = "http://maven.apache.org/POM/4.0.0
4  http://maven.apache.org/xsd/maven-4.0.0.xsd">
5     <modelVersion>4.0.0</modelVersion>
6
7     <groupId>com.companyname.project-group</groupId>
8     <artifactId>project</artifactId>
9     <version>1.0</version>
10 </project>
11
```

Introduction to POM File

- **Project tag** is the **root** of the file
- It should reference a basic schema settings such as **apache schema** and **w3.org** specification
- **Model version** describes the **version of Maven**
- **Group Id** is the id of the project's group (Simply it shows the **company** or the **organization** or the **owner of the project**)

```
1  <project xmlns = "http://maven.apache.org/POM/4.0.0"
2  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
3  xsi:schemaLocation = "http://maven.apache.org/POM/4.0.0
4  http://maven.apache.org/xsd/maven-4.0.0.xsd">
5     <modelVersion>4.0.0</modelVersion>
6
7     <groupId>com.companyname.project-group</groupId>
8     <artifactId>project</artifactId>
9     <version>1.0</version>
10 </project>
11
```

Introduction to POM File

- **Group Id** should be long enough to give **uniqueness** to the project
- **Artifact id** is the id for specifying the project under the group
- It shows the **name of the project** like pet-clinic-server
- **Version** defines the version number of the project

```
1  <project xmlns = "http://maven.apache.org/POM/4.0.0"
2  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
3  xsi:schemaLocation = "http://maven.apache.org/POM/4.0.0
4  http://maven.apache.org/xsd/maven-4.0.0.xsd">
5     <modelVersion>4.0.0</modelVersion>
6
7     <groupId>com.companyname.project-group</groupId>
8     <artifactId>project</artifactId>
9     <version>1.0</version>
10 </project>
11
```

Super POM

- Super POM is Maven's **default POM**
- **All POMs extend** the Super POM **unless explicitly set**
- Super POM and project POM creates the **Effective POM**
- Which is the **overall configuration** file
- Effective POM can be examined by running
  “mvn help:effective-pom”

Introduction to Build Lifecycles

- There are **three built-in lifecycles :**
  - **default**, **clean**, and **site**
  - Default is the **main lifecycle**
  - Clean is used for **cleaning the project**
  - Site lifecycle is used for building the **project's website**

Introduction to Build Lifecycles

- Each life cycle has a different number of phases
  - ✓ Default build lifecycle has **23 phases**
  - ✓ Clean lifecycle has **3 phases**
  - ✓ Site lifecycle has **4 phases**

Introduction to Build Lifecycles

Using Command-Line :

- Maven CLI commands generates your outputs
- For example,
  - ✓ "**mvn package**" gives you a "**jar, war or ear …**"
  - ✓ "**mvn test**" gives your test code's results
  - ✓ "**mvn clean**" cleans the artifacts of a previous command

Clean Lifecycle

Clean Lifecycle has **three phases**

- **pre-clean**, **clean**, and **post-clean**
- These phases are **in sequence**
- When a phase is called (for example "mvn post-clean"), phases prior to that phase are also run

It cleans the project's target directory

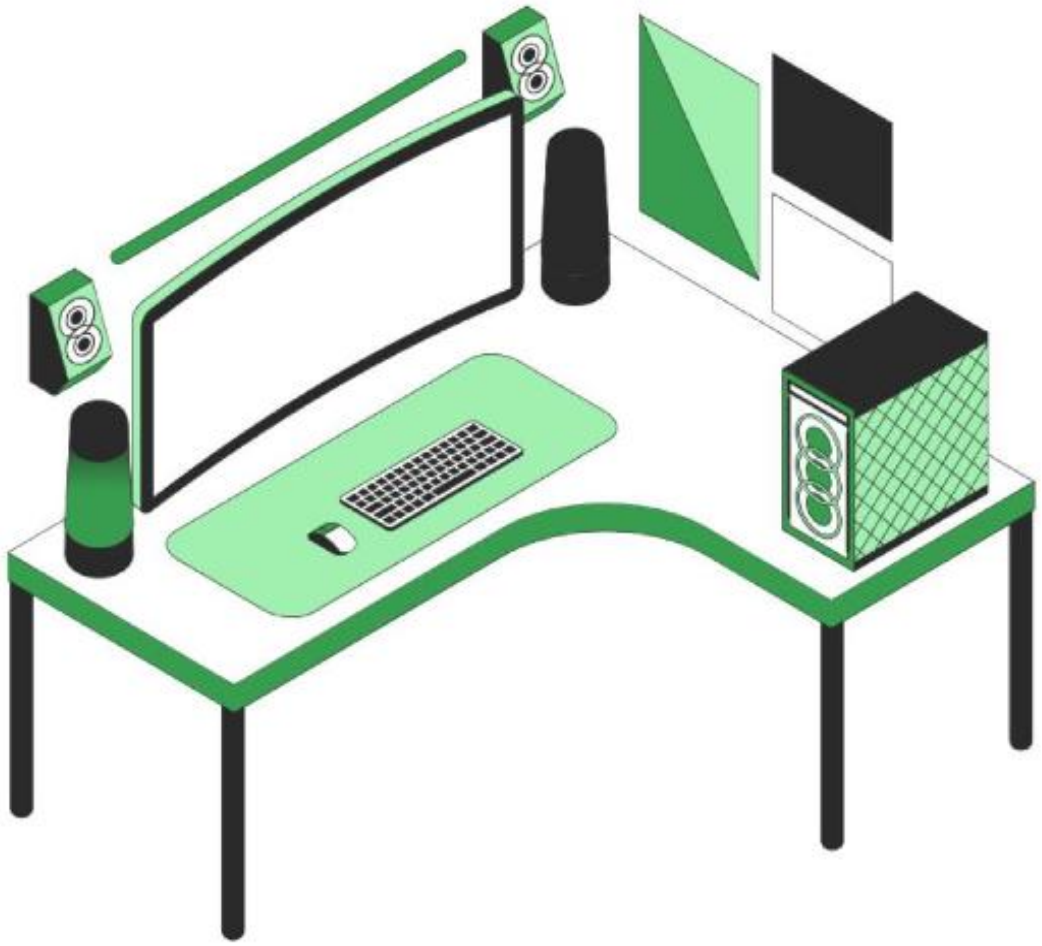
- **Pre-clean** phase is used for **any task prior to** the cleanup
- **Post-clean** phase is used for **any task following** the cleanup

Default Lifecycle

- Default lifecycle is used **for application build**
- There are **23 phases** in Default Lifecycle
- The most important phases are :
  - **validate:** validates if the project has necessary information
  - **compile:** compiles the source code
  - **test-compile:** compiles the test source code
  - **test:** runs unit tests
  - **package:** packages compiled source code
    - **packaging tag** in POM.xml changes the output
  - **integration-test:** processes and deploys the package if
  - needed to run integration test
  - **install:** installs the package to local repository
  - **deploy:** copies the package to a remote repository

Site Lifecycle

- Site lifecycle has **four phases**
  - **pre-site**, **site**, **post-site**, **site-deploy**
- For Site Lifecycle, the **Site Plugin is used**
- The plugin's **main duty** is to **generate a website**



Do you have any questions?

Send it to us! We hope you learned something new.