Day 3

Wednesday, July 27, 2022 3:42 PM

Maven

dependency manager and build automation tool for Java
- Create compiled jars/wars
  - Java Archive or Web Archive
  - Shippable java applications
- pull in dependencies without manually updating the class path

Maven project configuration and dependency management is done in the Project Object Model

AKA our pom.xml

The first thing you will encounter in the pom are the project coordinates
- how maven identifies a project

1. group-id: for example com.example
2. artifact-id: project name
3. version: 0.0.1-SNAPSHOT
   - this uniquely identifies the project version

More about our Pom
- Inside our Pom we will see tags which describe our project, and any build configurations

Pom tags:
- project: root tag of the Pom
- model Version: which version of the Pom
- name: name of the project
- properties: project specific properties
- dependencies: parent tag to list the projects dependencies
- dependency: individual dependencies you want for the project

- dependency: individual dependencies you want for the project

Maven Repositories

When Maven builds a project it must search for any dependencies declared in the POM.xml

It will look for these in two locatrons
- first locally $HOME/.m2/repository
- then it grab it from the central Maven repository
  mvnrepository.com

Maven LifeCycle

When Maven builds a project, it will take all the source code, all the dependencies, and compile it into a runnable artifact
- Javas version of an .exe
- .jar, .war, .ear
- Once created your application can run on a desktop or on a server accessible to others

Three main build life cycles
- Defualt handle project deployment
- Clean: handles project cleaning
- Site: handle the creation of project site documentation

The phases of the defualt Maven build life cycle
1. Validate: all needed info is available
2. Compile: java code → byte code
3. Test: test all compiled code
4. Package: creates the archive file .jar or .war
5. Integration: runs integration tests
6. Verify: checks resolts of previous step
7. Install: installs the jar/war into your local Maven repo
8. Deploy: copy the final jar/war to the remote repo