Gradle

Idaho State University | Computer Science

Dr. Isaac Griffith

CS 2263
Department of Computer Science
Idaho State University

ROAR

Outcomes

After today's lecture you will be able to:

- Understand why we use build and dependency management tools
- Understand the basics of gradle
- Initiate a gradle project
- Utilize the basic gradle tasks to build a java project
- Configure a gradle project

Automated Build in Practice

- In the prior lecture we learned about the concepts surrounding
  - Dependency Management
  - Automated Build
  - Continuous Integration

Automated Build in Practice

- In the prior lecture we learned about the concepts surrounding
  - Dependency Management
  - Automated Build
  - Continuous Integration

But, how do we put this into practice?

CS 2263

ROAR

What is Gradle?

- Gradle is a general purpose **build system**
- It comes with a rich build description language (DSL) based on **Groovy**
- It supports “**build-by-convention**” principle
- But it is very **flexible** and **extensible**
- It has **built-in plug-ins** for Java, Groovy, Scala, Web, OSGi
- It derives all the best and integrates well with **Ivy, Ant and Maven**

What is Gradle?

- Gradle is also a dependency management system
- It downloads required libraries (with specific versions) for use in your project.
- Gradle is similar to other tools used in other languages
  - Python has `pip`
  - JavaScript has `npm`
  - C# has `nuget`
  - C++ has `cmake` and `conan`
  - Ruby has `bundler`

Gradle Features

- Declarative builds and build-by-convention
- Language for dependency based programming and many ways to manage dependencies
- Groovy as a base langauge allows imperative programming

Gradle Features

- Deep and rich API for managing projects, tasks, dependency artifacts and much more
- State of the art support for multi-project builds
- Ease of integration and migration
- Free and open source

Advanced Features

- Parallel unit test execution
- Dependency build
- Incremental build support
- Dynamic tasks and task rules
- Gradle daemon

Using Gradle

CS 2263

ROAR

A Basic Java Project

build.gradle file

```
plugins {
  id 'java'
}

repositories {
  mavenCentral()
}

dependencies {
  testRuntime "org.junit.jupiter:junit-jupiter-engine:5.5.2"
  testRuntime "org.junit.platform:junit-platform-runner:1.5.2"
}

test {
  useJUnitPlatform()
}
```

Directory Structure

project_root/

- src
  - main
    - java
    - resources
  - test
    - java
    - resources
- build.gradle
- settings.gradle

Java Application Project

build.gradle file

```
plugins {
  id 'java'
  id 'application'
}

repositories {
  mavenCentral()
}

dependencies {
  testRuntime "org.junit.jupiter:junit-jupiter-engine:5.5.2"
  testRuntime "org.junit.platform:junit-platform-runner:1.5.2"
}

mainClass = 'App'

test {
  useJUnitPlatform()
}
```

Directory Structure

project_root/

- app/src
  - main
    - java
    - resources
  - test
    - java
    - resources
- app/build.gradle
- settings.gradle

Idaho State University
Computer Science

Java Library Project

build.gradle file

```
plugins {
  id 'java-library'
}

repositories {
  mavenCentral()
}

dependencies {
  testRuntime "org.junit.jupiter:junit-jupiter-engine:5.5.2"
  testRuntime "org.junit.platform:junit-platform-runner:1.5.2"
}

test {
  useJUnitPlatform()
}
```

Directory Structure

project_root/

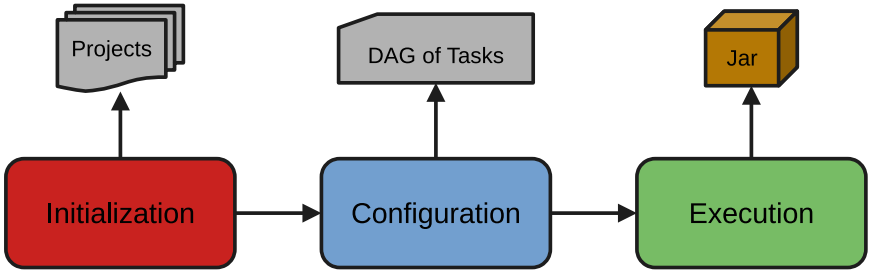
- lib/src
  - main
    - java
    - resources
  - test
    - java
    - resources
- lib/build.gradle
- settings.gradle

ROAR

Core Concepts

- **Build Script**: a build configuration script supporting one or more project
- **Project**: a component that needs to be built. It is made up of one or more tasks
- **Task**: a distinct step required to perform the build. Each task/step is atomic (either succeeds or fails).
- **Publication**: the artifact produced by the build process

Dependency Resolution



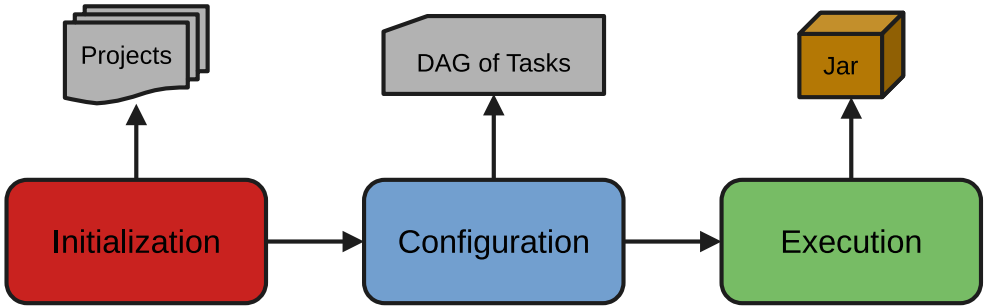
- **Dependencies**: tasks and projects depending on each other (internal) or on third-party artifacts (external).
- **Transitive dependencies**: the dependencies of a project may themselves have dependencies
- **Repositories**: the “places” that hold external dependencies (Maven/Ivy repos, local folders).
- **DAG**: the directed acyclic graph of dependencies (what depends on what)
- **Dependency configurations**: named sets (groups) of dependencies (e.g. per task)

Plugins

- A plugin applies a set of extensions to the build process.
  - Add tasks to a project
  - Pre-configure these tasks with reasonable defaults.
  - Add dependency configurations
  - Add new properties and methods to existing objects
- Plugins implement the “build-by-convention” principle in a flexible way

The Build Lifecycle

1. **Initialization**: initialization of the project
2. **Configuration**: configuration of the project (computes the DAG)
3. **Execution**: executes the sequence of build tasks



A Simple Example

CS 2263

ROAR

Initiating a Project

- To initilize a project as a gradle project, you need to:
  - include a “build.gradle” in the root project directory
  - setup the proper directory structure
- Alternatively, you can let gradle do this for you by
  - Executing the following in the root project directory

```
> gradle init
```

Run a build task

```
> gradle test
```

Compiles the source and runs the tests

```
> gradle tasks
```

clean, assemble, build, classes, testClasses, test, jar, etc

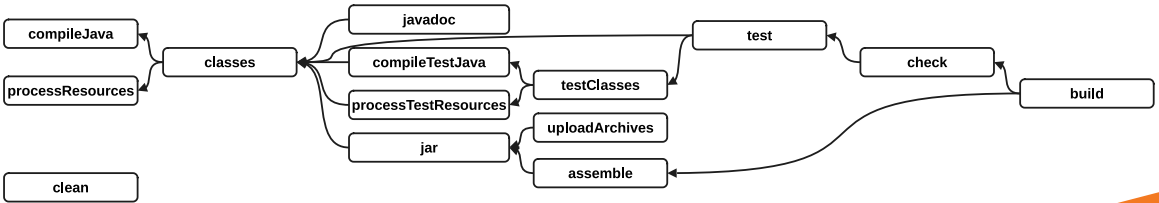
Standard Java Tasks

Tasks added by Java Plugin

- `compileJava`
- `jar`
- `javadoc`
- `clean`
- `test`

Lifecycle Tasks

- `assemble`
- `check`
- `build`



Another Example

CS 2263

ROAR

Dependency Mgmt

CS 2263

ROAR

Repository Configuration

```
repositories {
  mavenCentral()
}
```

```
{
  mavenCentral name: 'single-jar-repo', urls: "http://repo.mycompany.com/jars"
  flatDir name: 'localRepository',
  dirs: 'lib' flatDir dirs: ['lib1', 'lib2']
}
```

Referencing Dependencies

- **General Syntax**
  - `<configuration> '<reference-string>'`
  - `<configuration> group: '<group-name>', name: '<artifact-name>', version: '<version>'`

Example:

```
dependencies {
  testImplementation 'junit:junit:4.7'
  implementation group: 'org.springframework', name: 'spring-core', version: '2.5'
}
```

Finding Dependencies

- If you already know the library name
  - Just search MavenCentral repository
  - **Example…**
- Otherwise,
  - This may require a more involved google search
  - Review of project documentation for installation using gradle/maven

Dependency Configurations

- Plugins like `java` and `groovy` have predefined dependency configurations, but you may also create your own

```
configurations {
  foobar
}

dependencies {
  foobar 'junit:junit:4.7'
}
```

Built-in Java Configurations

- `implementation` - implementation only dependencies
- `compileOnly` - compile time only dependencies, not used at runtime
- `compileClasspath` - compile classpath, used when compiling source. Used by task `compileJava`
- `annotationProcessor` - annotation processors used during compilation
- `runtimeOnly` - runtime only dependencies
- `runtimeClasspath` - runtime classpath contains elements of the implementation, as well as runtime only elements

Built-in Java Configurations

- `testImplementation` - implementation only dependencies for tests
- `testCompileOnly` - additional dependencies only for compiling tests, not used at runtime
- `testCompileClasspath` - test compile classpath, used when compiling test sources. Used by task `compileTestJava`
- `testRuntimeOnly` - runtime only dependencies for running tests
- `testRuntimeClasspath` - runtime classpath for tunning tests. Used by task `test`
- `archives` - artifacts (e.g., jars) produced by this project. Used by task `uploadArchives`

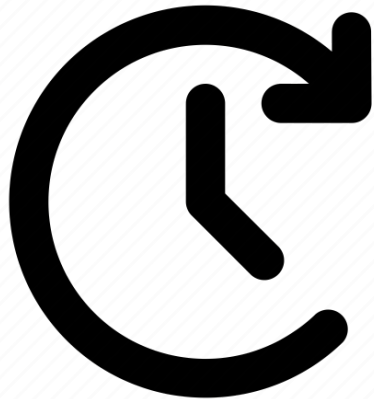
Gradle & Your IDE

CS 2263

ROAR

For Next Time

- Review the Gradle Readings
- Review this Lecture
- Come to Class
- Continue working on Homework 01
  - It's Due Sunday at 23:00
- Read Git Book Chapter 1
- Read the What is Version Control article



Before class make sure you have installed the latest version of Gradle and can run it from the command line.

Are there any questions?