MÓDULO 1 - EXTRA
INTRODUÇÃO AO GRADLE

UNIPDS
EDUCAÇÃO

Por que outra ferramenta de build?

Ant, o antigo

build.xml

```xml
<project>
    <target name="clean">
        <delete dir="classes" />
    </target>
    <target name="compile" depends="clean">
        <mkdir dir="classes" />
        <javac srcdir="src" destdir="classes" />
    </target>
    <target name="jar" depends="compile">
        <mkdir dir="jar" />
        <jar destfile="jar/hello.jar" basedir="classes">
            <manifest>
                <attribute name="Main-Class"
                  value="br.com.unipds.HelloWorld" />
            </manifest>
        </jar>
    </target>
</project>
```

- Lançado em 2000
- Formato XML
- Programático, baseado em tasks
- Copia e cola para novos projetos
- Não gerencia dependências (jars), só com Ivy, um módulo adicional.

```
$ ant jar
```

jar ⟶ compile ⟶ clean

Maven, o popular

pom.xml

```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>br.com.unipds</groupId>
    <artifactId>hello</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

- Lançado em 2004
- Formato XML
- Declarativo, com fases pré-definidas
- Pensado a partir de projetos reais da Apache Software Foundation
- Gerenciamento de dependências nativo

```
$ mvn package
```

package → test → test-compile → compile → validate

Tarefas complexas no Build

E se eu precisar definir uma lógica complexa para alguma tarefa do build?

Exemplo de um projeto real: definir o nome de um DB schema para testes cujo nome é derivado do nome da máquina e um timestamp

- Não dá pra programar essa lógica em XML
- No Maven, teríamos que criar uma classe Java, compilar, etc...

Plugins do Maven

```java
@Mojo(name = "generate-test-schema", defaultPhase = LifecyclePhase.INITIALIZE)
public class GenerateDbNameMojo extends AbstractMojo {

    @Parameter(defaultValue = "${project}", readonly = true, required = true)
    private MavenProject project;

    @Parameter(property = "schemaNameProperty", defaultValue = "test.db.schema.name")
    private String schemaNameProperty;

    @Override
    public void execute() throws MojoExecutionException {
        try {
            String cleanHostname = InetAddress.getLocalHost()
                .getHostName().replaceAll("[^\\w]", "");
            String generatedName = String.format("%s_%s_%d", project.getArtifactId(),
                cleanHostname, new Date().getTime());

            Properties projectProperties = project.getProperties();
            projectProperties.setProperty(schemaNameProperty, generatedName);
        } catch (UnknownHostException e) {
            throw new MojoExecutionException("Falha ao resolver o hostname", e);
        }
    }
}
```

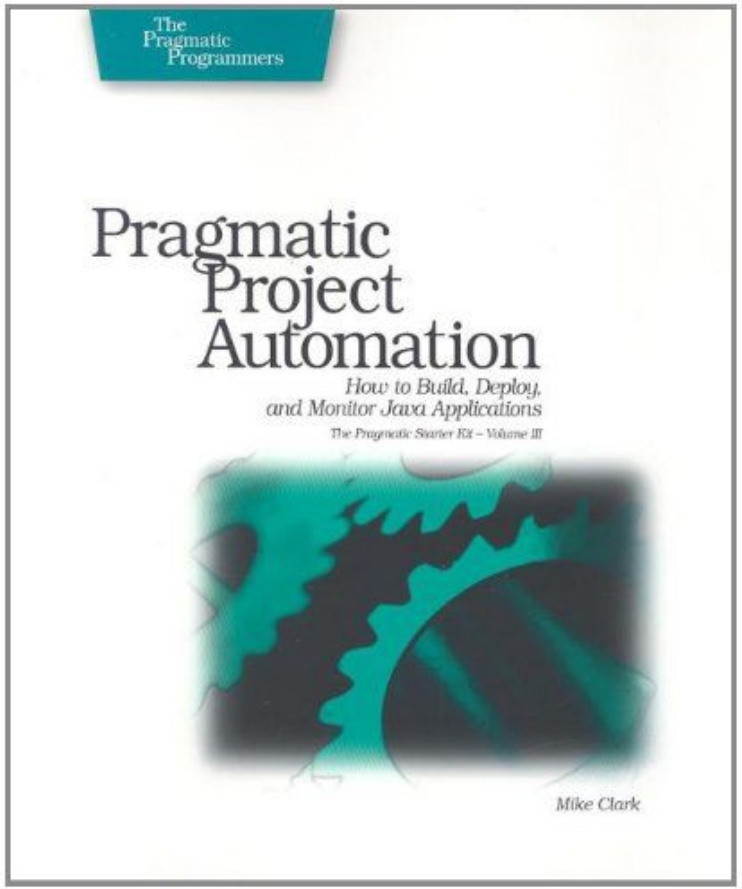
- Verboso
- Difícil
- Fora do build

XML e o erro do criador do Ant

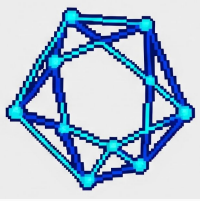
“XML provavelmente **não foi a escolha correta** como parecia”

“Minha intenção **nunca** foi que o formato de arquivo se transformasse em um linguagem de script.”

“Se eu soubesse o que sei agora, eu **teria tentado usar uma linguagem de script de verdade**, como JavaScript via o componente Rhino ou Python via Jython, com bindings para objetos Java”



Gradle, o moderno



build.gradle

```
plugins {
  id 'java'
}

group = 'br.com.unipds'
version = '1.0-SNAPSHOT'

repositories {
  mavenCentral()
}

dependencies {
  testImplementation platform('org.junit:junit-bom:5.10.0')
  testImplementation 'org.junit.jupiter:junit-jupiter'
}

test {
  useJUnitPlatform()
}
```

- Lançado em 2007
- Scripts Groovy ou Kotlin
- Declarativo e programático
- Gerenciamento de dependências nativo, reusa repositórios Maven

```
$ gradle build
```

build → assemble → jar → classes → compileJava

Lógica no build do Gradle

```
import java.net.InetAddress

ext {
    cleanHostname = InetAddress.getLocalHost().getHostName().replaceAll("[^a-zA-Z0-9_\\-]", "_")
    testDbSchemaName = "${project.name}_${cleanHostname}_${new Date().getTime()}"
}

tasks.named('test') {
    systemProperty 'test.db.schema.name', project.ext.testDbSchemaName
}
```

- Sucinto
- Legível
- Dentro do build

Iniciando com Gradle

Como instalar?

Pré-requisito: JDK instalada

CLI

```
$ gradle build
```

- Baixar o zip em https://gradle.org/releases/
- Instalador como `sdk`, `brew`, etc

Gradle Wrapper

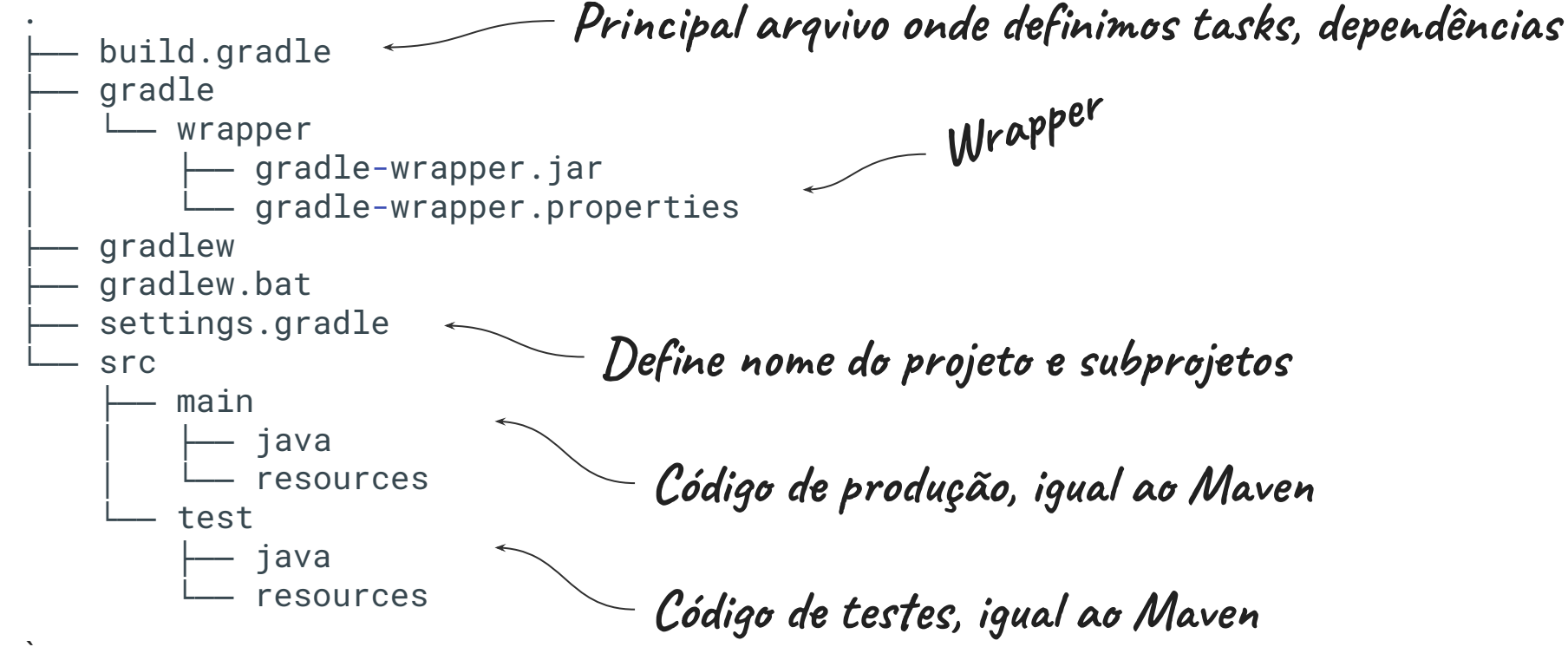
```
$ ./gradlew build
```

mais comum

- Criado pela IDE (em geral)
- Contém scripts mínimos Shell e batch (Windows) que baixam o Gradle

```
.
├── gradle
│   └── wrapper
│       ├── gradle-wrapper.jar
│       └── gradle-wrapper.properties
├── gradlew
└── gradlew.bat
```

Estrutura de arquivos



build.gradle

```
plugins {
    id 'java'
}

group = 'mx.florinda'
version = '1.0-SNAPSHOT'

repositories {
    mavenCentral()
}

dependencies {
    testImplementation platform('org.junit:junit-bom:5.10.0')
    testImplementation 'org.junit.jupiter:junit-jupiter'
}

test {
    useJUnitPlatform()
}
```

Provê suporte a projetos Java (compileJava, jar, javadoc)

Equivalem a groupId e version do Maven

Reutiliza o repositório do Maven para baixar dependências

Define dependências a serem baixadas

Configurações de testes

settings.gradle

```
rootProject.name = 'cardapio'
```

./gradlew tasks

```
> Task :tasks

------------------------------------------------------------
Tasks runnable from root project 'cardapio'
------------------------------------------------------------

Build tasks
-----------
assemble - Assembles the outputs of this project.
build - Assembles and tests this project.
buildDependents - Assembles and tests this project and all projects that depend on it.
buildNeeded - Assembles and tests this project and all projects it depends on.
classes - Assembles main classes.
clean - Deletes the build directory.
jar - Assembles a jar archive containing the classes of the 'main' feature.
testClasses - Assembles test classes.

....

BUILD SUCCESSFUL in 464ms
1 actionable task: 1 executed
```

src/main/java/mx/florinda/cardapio/ItemCardapio.java

```java
package mx.florinda.cardapio;

import java.math.BigDecimal;

public record ItemCardapio(Long id, String nome, String descricao,
                           ItemCardapio.CategoriaCardapio categoria,
                           BigDecimal preco, BigDecimal precoPromocional) {

    public static enum CategoriaCardapio {
        ENTRADAS, PRATOS_PRINCIPAIS, BEBIDAS, SOBREMESA;
    }

}
```

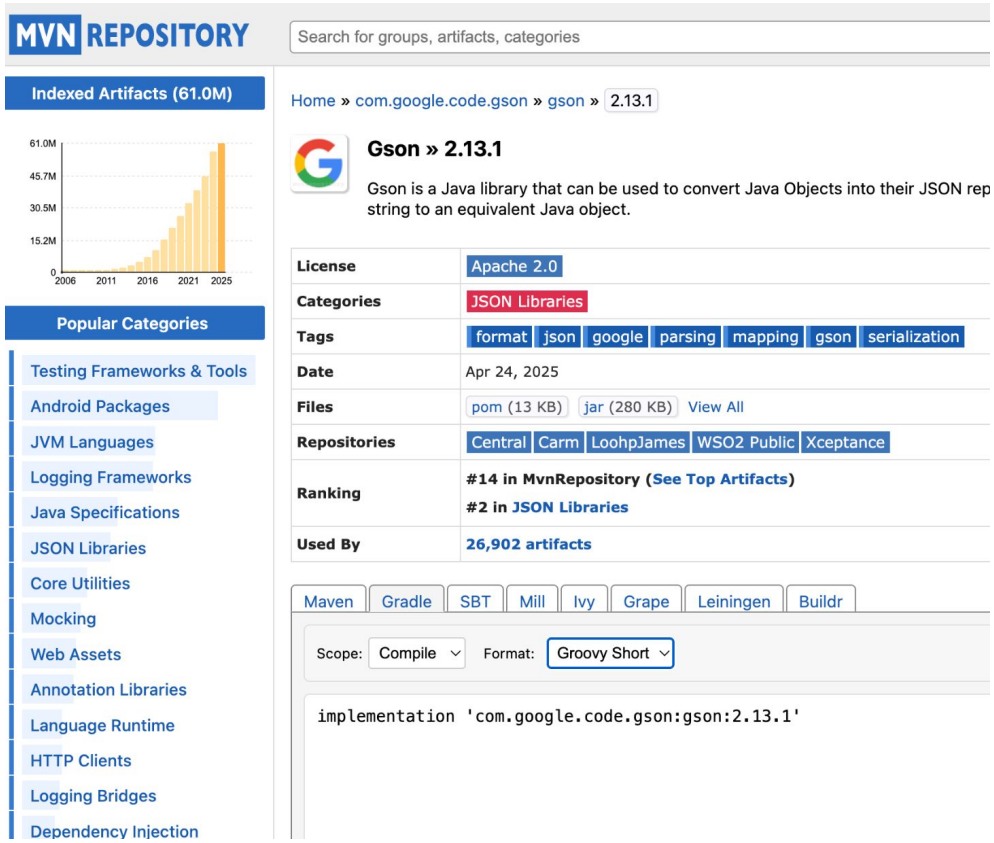
```
$ ./gradlew build

BUILD SUCCESSFUL in 477ms
2 actionable tasks: 2 up-to-date
```

Gera

build/classes *com* .class *e*

build/libs *com* .jar



As dependências ficam aqui

```
~/.gradle/caches/modules-2/files-2.1
.
├── com.google.code.gson
│   ├── gson
│   │   └── 2.13.1
│   │       ├── 75c68fafbbc6c1abc95a6...
│   │       │   └── gson-2.13.1.pom
│   │       └── 853ce06c11316b33a8eae...
│   │           └── gson-2.13.1.jar
├── com.google.errorprone
├── org.apiguardian
├── org.junit.jupiter
├── org.junit.platform
└── org.opentest4j
```

- **Groovy Short:** `implementation 'com.google.code.gson:gson:2.13.1'`
- **Groovy Long:** `implementation group: 'com.google.code.gson', name: 'gson', version: '2.13.1'`
- **Kotlin:** `implementation("com.google.code.gson:gson:2.13.1")`

https://mvnrepository.com/artifact/com.google.code.gson/gson/2.13.1

./gradlew dependencies --configuration runtimeClasspath

```
> Task :dependencies

------------------------------------------------------------
Root project 'florinda'
------------------------------------------------------------

runtimeClasspath - Runtime classpath of source set 'main'.
\--- com.google.code.gson:gson:2.13.1
     \--- com.google.errorprone:error_prone_annotations:2.38.0

A web-based, searchable dependency report is available by adding the --scan option.

BUILD SUCCESSFUL in 468ms
1 actionable task: 1 executed
```

src/main/java/mx/florinda/carpadio/Main.java

```java
import com.google.gson.Gson;
import java.math.BigDecimal;
import static mx.florinda.cardapio.ItemCardapio.CategoriaCardapio.*;

public class Main {
    public static void main(String[] args) {
        var refresco = new ItemCardapio(1L, "Refresco do Chaves",
          "Suco de limão, que parece tamarindo mas tem gosto de groselha",
          BEBIDAS, new BigDecimal("2.99"), null);

        Gson gson = new Gson();
        String json = gson.toJson(refresco);
        System.out.println(json);
    }
}
```

Usando o Gson