Programação Avançada - AULA Maven

Matheus Moresco

Análise e Desenvolvimento de Sistemas - 5º Período

2025/01

O que é o Maven?

- O **Apache Maven** é uma **ferramenta de automação de build** usada principalmente em projetos Java. Ele ajuda os desenvolvedores a **compilar, testar, empacotar e gerenciar dependências** de forma automática e padronizada.

Problemas em projetos sem gerenciador de build

1. Gerenciamento Manual de Dependências
2. Compilação e Execução Complicadas
3. Falta de Padronização na Estrutura de Projeto
4. Builds Inconsistentes
5. Dificuldade em Automatizar Tarefas
6. Testes Manuais ou Negligenciados
7. Integração Contínua Prejudicada

O que o Maven faz?

- **Gerencia dependências**
  - Você declara as bibliotecas que quer no arquivo pom.xml, e o Maven baixa tudo automaticamente do repositório central (como o Maven Central).
- **Automatiza o processo de build**
  - Compilação (mvn compile)
  - Testes (mvn test)
  - Geração de JARs ou WARs (mvn package)
  - Deploy em servidores ou repositórios (mvn deploy)
- **Define uma estrutura padrão de projeto**
- **Facilita a integração com ferramentas de CI/CD**
  - Jenkins, GitHub Actions, GitLab CI, etc.

Configuração do Maven

- O Maven usa um arquivo chamado pom.xml (Project Object Model), que contém:
  - Informações do projeto (nome, versão, etc.)
  - Dependências (bibliotecas que o projeto usa)
  - Plugins (ferramentas extras para compilar, empacotar, etc.)
  - Etapas de build

```xml
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.exemplo</groupId>
  <artifactId>meuprojeto</artifactId>
  <version>1.0.0</version>

  <dependencies>
    <dependency>
      <groupId>com.google.code.gson</groupId>
      <artifactId>gson</artifactId>
      <version>2.10.1</version>
    </dependency>
  </dependencies>
</project>
```

Arquivo pom.xml

O arquvi pom.xml serve para definir as configurações do maven, adicionar dependências do projeto e comportamentos personalizados.

- **Dependencies**: lista as **bibliotecas que seu projeto usa**.
- **GroupId**, **artifactId**, **version**: informações da biblioteca que será adicionada ao projeto.
- **Build** (**plugins**): usado quando você quer executar a aplicação diretamente pelo Maven, criar um .jar executável ou customizar o processo de build, você adiciona plugins aqui.

Instalando o Maven

O Maven costuma ser instalado automaticamente com IDEs como o Eclipse e o netBeans. Porém, caso você não tenha o Maven instalado, você precisa:

- Acessar o site do Maven para seguir o passo a passo da instalação

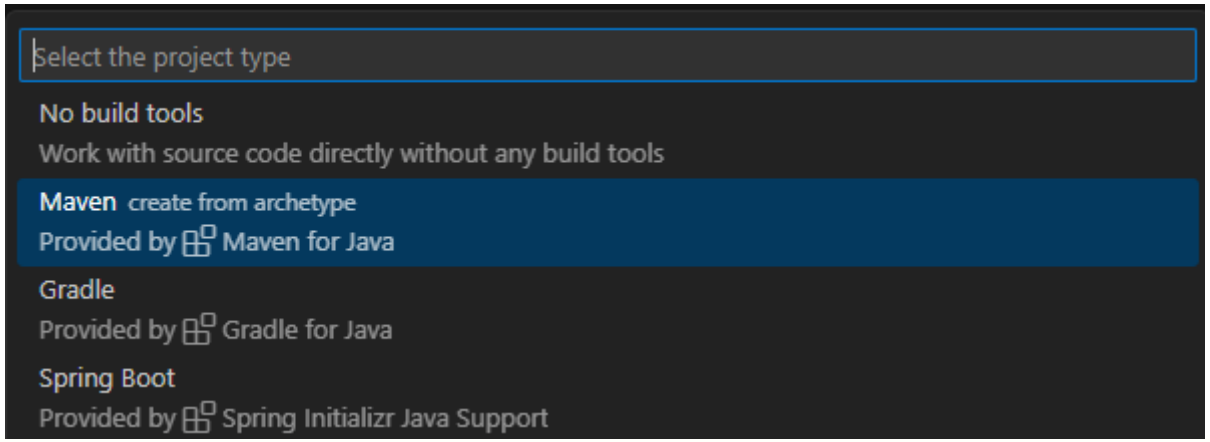
Criando um projeto Maven

- É possível criar um projeto Maven de maneiras diferentes:

1. Pelo terminal, através de linha de comando:

```
mvn archetype:generate -DgroupId=com.exemplo -DartifactId=meuprojeto -
DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

2. Através da IDE:



Archetypes do Maven

No Maven, os archetypes são como modelos prontos de projeto. Eles geram a estrutura básica de um projeto com os arquivos e diretórios necessários para você começar a desenvolver, sem precisar montar tudo na mão.

- Estrutura de pastas (ex: src/main/java, src/test/java)
- Arquivo pom.xml básico
- Classe Main e classe de teste

Principais Archetypes do Maven

Dentre os archetypes do Maven, podemos destacar os seguintes:

- **maven-archetype-quickstart** : O mais usado para projetos Java simples
- **maven-archetype-webapp** : Para criar aplicações web Java (Servlets, JSP)
- **maven-archetype-archetype** : Para criar o seu *próprio archetype* personalizado
- **maven-archetype-site-simple** : Cria um projeto de site de documentação com Maven Site Plugin.
- **spring-boot-archetypes** : padrão de um archetype oficial para Spring Boot.
- Entre outros.

Exemplo prático

- Criar um Projeto simples com Maven
- Adicionar dependências.