(✓) Curso Técnico

Quem Investe no futuro faz QI

Informática para Internet

Desenvolvimento de Sistemas

Web III

Unidade VIII

1. Gerenciador de dependências

A evolução na criação, programação e no desenvolvimento dos softwares, tornou impraticável o que há alguns anos era muito comum. Exemplo deste aspecto são os projetos que utilizam os componentes prontos e frameworks para funcionar.

Componentes, elementos e frameworks são dependências que precisam ser gerenciadas de alguma forma. Importante entender que outros tipos de dependências poderão ser utilizadas num software, aumentando a complexidade de gerenciamento, como um jQuery.

Os gerenciadores de dependências são utilizados para gerenciar as ações relacionadas as dependências de um projeto de software, permitindo listar, adicionar, remover, atualizar e procurar por dependências e mesmo identificar compatibilidade entre versões.

Quando se fala da Linguagem Java, a construção de um projeto de software geralmente consiste em tarefas como baixar dependências, colocar jars adicionais em um caminho de classe, compilar o código-fonte em código binário, executar testes, empacotar o código compilado em artefatos implantáveis, como arquivos JAR, WAR e ZIP, e implantar esses artefatos para um servidor de aplicativos ou repositório.

É possível automatizar estes processos e procedimentos, utilizando uma ferramenta de gerenciamento.

Vale salientar que um arquivo XML detentor do conceito POM[1] - Project Object Model, é uma super ferramenta de descrição, construção e gerenciamento de projetos de software Java usando informação centralizada.

O Apache Maven é uma ferramenta de automação e gerenciamento de projetos que fornece uma forma padronizada de automação, construção(build) e

[1] POM é a unidade fundamental de trabalho no Maven. É um arquivo XML que contém informações sobre o projeto e detalhes de configuração usados pelo Maven para construir o projeto. Ele contém valores padrão para a maioria dos projetos. Exemplos disso são o diretório de compilação, que é target; o diretório de origem, que é src/main/java; o diretório de origem do teste, que é src/test/java; e assim por diante. Ao executar uma tarefa ou objetivo, o Maven procura o POM no diretório atual. Ele lê o POM, obtém as informações de configuração necessárias e executa a meta.
Algumas das configurações que podem ser especificadas no POM são as dependências do projeto, os plugins ou objetivos que podem ser executados, os perfis de construção e assim por diante. Outras informações como a versão do projeto, descrição, desenvolvedores, listas de discussão e outras também podem ser especificadas.

publicação de aplicações. O Maven é uma ferramenta bastante flexível permitindo estender suas funcionalidades nativas através da adição de plugins.

Disto isto, o Apache Maven é uma ferramenta que permite automatizar essas tarefas, minimizando o risco de erros humanos durante a construção manual do software, separando o trabalho de compilação e empacotamento da construção do código, ou seja, é uma ferramenta poderosa de gerenciamento de projetos baseada no Project Object Model, que auxilia no gerenciamento de compilações de projetos, documentação, dependência, lançamentos dentre outros.

Outra ferramenta de aceleração e facilitação da manutenção de projetos de software Java é o Apache Ant.

Este é uma biblioteca Java e ferramenta de linha de comando cujo propósito é conduzir processos descritos em arquivos de construção como alvos e pontos de extensão dependentes uns dos outros. O principal uso conhecido do Ant é a construção de aplicativos Java. O Ant fornece várias tarefas integradas que permitem compilar, montar, testar e executar aplicativos Java. O Ant também pode ser usado efetivamente para construir aplicativos não Java, por exemplo, aplicativos C ou C++. De forma mais geral, o Ant pode ser usado para pilotar qualquer tipo de processo que possa ser descrito em termos de alvos e tarefas.

Ant é escrito em Java. Os usuários do Ant podem desenvolver seus próprios "antlibs" contendo tarefas e tipos do Ant, e são oferecidos um grande número de "antlibs" comerciais ou de código aberto prontos.

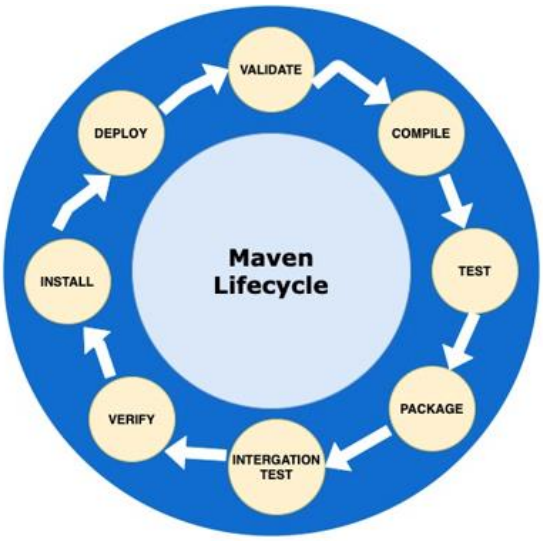
O Ant é extremamente flexível e não impõe convenções de codificação ou layouts de diretórios aos projetos Java que o adotam como ferramenta de construção.

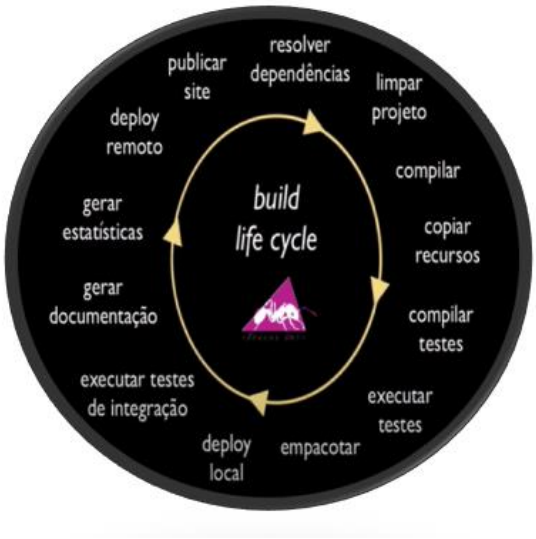
Os projetos de desenvolvimento de software que procuram uma solução que combine ferramentas de construção e gerenciamento de dependências podem usar o Ant em combinação com o Apache Ivy.

Vale analisar as diferenças entre as duas ferramentas citadas:

	Maven	Ant
Definição	É um framework baseado no conceito de POM.	É uma biblioteca Java e uma caixa de ferramentas de linha de comando.
Convenção	Possui convenções integradas para colocar código-fonte, código compilado, etc.	Não possui convenções formais.
Informações sobre a estrutura do projeto	Não requer que informações sobre a estrutura do projeto sejam fornecidas no arquivo pom.xml.	Requer que informações sobre a estrutura do projeto sejam fornecidas no arquivo build.xml.
Vida útil	Tem um ciclo de vida.	Não tem um ciclo de vida.
Natureza	É de natureza declarativa (apenas a fonte deve estar presente no diretório padrão).	É de natureza procedimental (diga manualmente exatamente o que fazer e quando fazer).
Modelo	É principalmente uma ferramenta de gerenciamento de projetos.	É principalmente uma ferramenta de gerenciamento de projetos.
Dependência	Ele pode baixar automaticamente as dependências de um repositório central para projetos de construção.	Não tem suporte integrado para gerenciamento de dependências.
Reutilização	Ele consiste em plug-ins reutilizáveis.	Consiste em scripts que não são reutilizáveis.
Preferência	É menos preferido.	É mais preferido.

Complexidade	É mais complexo.	É simples e confiável.
Flexibilidade	É menos flexível e de fácil manutenção.	É mais flexível e de fácil manutenção.





Observa-se que as fases de validar(valida se o projeto está correto); compilar (constrói o projeto); testar (projeto passe nos testes de unidade);empacotar(empacota o código compilado e o transforma em algum formato .jar ou .war); integrar(projeto passe nos testes de integração); verificar(verifica se o código empacotado é válido e atende aos critérios de qualidade); instalar (empacota o código do repositório Maven local, para uso como dependência em outros projetos) e implantar (implanta o código em um ambiente), presentes no Maven estão de alguma forma presentes no Ant.

Além do Maven e Ant, há o Apache IVY, este uma ferramenta de gerenciamento de dependências semelhante ao Maven, mas é baseado em ANT. É altamente flexível e configurável semelhante ao ANT e fácil de usar como Maven.

Pode-se dizer que o Ivy é muito popular e muito poderoso, sendo utilizado para gerenciar dependências em projetos baseados em ANT da mesma forma que o Apache Maven.

O plug-in Ivy Publish fornece a capacidade de publicar artefatos de compilação no formato Apache Ivy, geralmente em um repositório para consumo por outras compilações ou projetos. O que é publicado é um ou mais artefatos criados pela

compilação e um descritor de módulo Ivy (normalmente ivy.xml) que descreve os artefatos e as dependências dos artefatos, se houver.

Vale ressaltar os recursos importantes do Apache Ivy:

- ✓ Baseado em ANT - Apache Ivy fornece uma capacidade de gerenciamento de dependências para projetos baseados em ANT. É muito simples de usar também.
- ✓ Relatórios de dependências - Apache Ivy fornece opções para imprimir gráfico de dependências em html, bem como em formato de relatórios.
- ✓ Não intrusivo − Apache Ivy não impõe nenhuma restrição para fazer parte da distribuição. Mesmo os arquivos de compilação não dependem do Apache Ivy.
- ✓ Altamente flexível - Apache Ivy fornece muitas configurações padrão e pode ser configurado de acordo com o requisito com muita facilidade.
- ✓ Extensível - Apache Ivy pode ser estendido facilmente. Você pode definir seu próprio repositório, resolvedores de conflitos e a estratégia mais recente.
- ✓ Desempenho - Apache Ivy é construído para desempenho. Ele mantém um cache da biblioteca já baixada. Procura primeiro nos repositórios locais para resolver as dependências do que em outros repositórios.
- ✓ Dependências transitivas - Apache Ivy gerencia automaticamente dependências transitivas se um projeto ou biblioteca depende de outra biblioteca que pode precisar de outra biblioteca.
- ✓ Repositório Maven - Apache Ivy segue convenções semelhantes às convenções do repositório Maven. O Apache Ivy pode resolver dependências usando o repositório global maven.
- ✓ Maven 2 POMs - Apache Ivy pode ler Maven 2 POMs como descritores de módulo, pode definir ivy como descritor de módulo. Assim, facilita a migração de projetos existentes para projetos gerenciados pela IVY.
- ✓ Publicação - Apache Ivy fornece suporte para publicar seu projeto e simplifica o processo de implantação do ambiente multiprojeto.
- ✓ Gratuito para usar - o Apache Ivy é de código aberto e de uso gratuito.
- ✓ Documentação - Apache Ivy tem uma documentação muito detalhada e tutoriais disponíveis para aprender.

Além do Maven, Ant e Ivy, há o Gradle, uma ferramenta de automação de compilação de código aberto flexível o suficiente para criar quase qualquer tipo de software. Valendo salientar que o Gradle quase não delimita ou supõe o que será construído/desenvolvido ou como será, o que o torna particularmente flexível.

O design no Gradle é baseado nos seguintes fundamentos:

Alta performance: O Gradle evita trabalho desnecessário executando apenas tarefas que precisam funcionar porque as entradas ou saídas foram alteradas. O Gradle usa vários caches para reutilizar as saídas de compilações anteriores. Com um cache de compilação compartilhado, você pode até reutilizar as saídas de outras máquinas.

Fundação da JVM: O Gradle é executado na JVM. Este é um bônus para usuários familiarizados com Java, já que a lógica de construção pode usar as APIs Java padrão. Também facilita a execução do Gradle em diferentes plataformas.

Convenções: O Gradle facilita a criação de tipos comuns de projetos por meio de convenções. Os plug-ins definem padrões sensatos para manter os scripts de construção mínimos. Mas essas convenções não o limitam: você pode definir configurações, adicionar suas próprias tarefas e fazer muitas outras personalizações em suas compilações.

Extensibilidade: A maioria das compilações tem requisitos especiais que exigem lógica de compilação personalizada. Você pode estender facilmente o Gradle para fornecer sua própria lógica de compilação com tarefas e plug-ins personalizados. Veja as compilações do Android para obter um exemplo: elas adicionam muitos novos conceitos de compilação, como variações e tipos de compilação.

suporte IDE: Vários IDEs importantes fornecem interação com compilações Gradle, incluindo Android Studio, IntelliJ IDEA, Eclipse, VSCode e NetBeans. O Gradle também pode gerar os arquivos de solução necessários para carregar um projeto no Visual Studio.

Entendimento: O Build Scan™ fornece informações abrangentes sobre uma compilação que você pode usar para identificar problemas. Você pode usar Build Scans para identificar problemas com o desempenho de uma compilação e até mesmo compartilhá-los para ajudar na depuração.

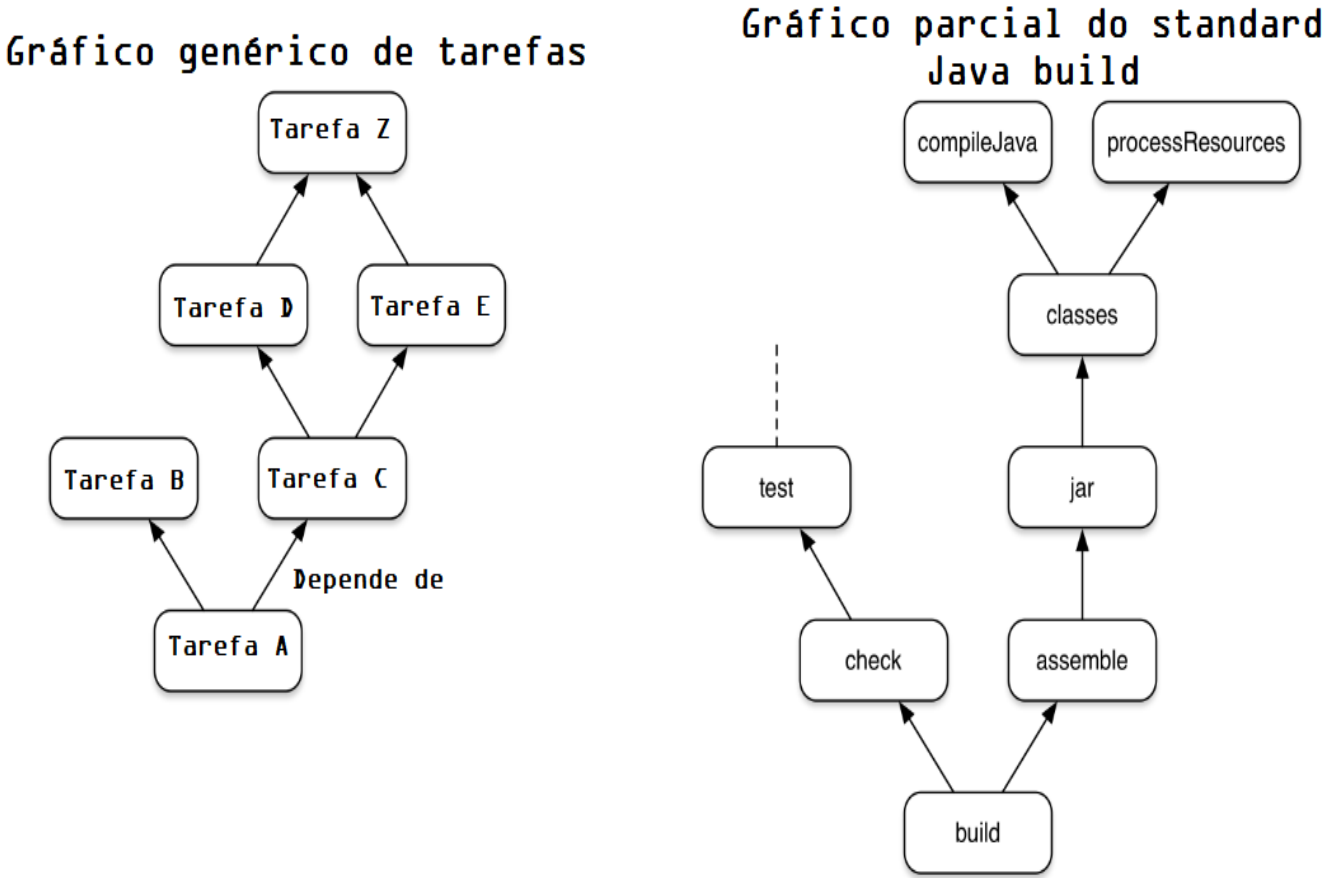
Gradle é um exemplo de programação baseada em dependência, pois possibilita definir tarefas e dependências entre tarefas. O Gradle garante que essas

tarefas sejam executadas na ordem de suas dependências. Seus scripts de construção e plug-ins configuram esse gráfico de dependência.

Algumas ferramentas de construção montam um gráfico de tarefa enquanto executam tarefas. Gradle constrói o grafo da tarefa antes de executar qualquer tarefa. Com a prevenção de configuração, o Gradle ignora a configuração de tarefas que não fazem parte da compilação atual.

Dentro de cada projeto, as tarefas formam um gráfico acíclico direcionado (DAG - Directed Acyclic Graph).

A seguir o diagrama mostra dois exemplos de gráficos de tarefas: um abstrato e outro concreto. O diagrama representa as dependências entre as tarefas com setas:



Descreve-se as fases do ciclo de vida pelo qual o Gradle passa ao interpretar scripts:

Fases de construção: Uma compilação Gradle tem três fases distintas. O Gradle executa essas fases em ordem: primeira inicialização, depois configuração e, finalmente, execução.

- Inicialização

    - Detecta o arquivo de configurações.
    - Avalia o arquivo de configurações para determinar quais projetos e compilações incluídas participam da compilação.
    - Cria uma *Project*[2] instância para cada projeto.
- Configuração
    - Avalia os scripts de construção de cada projeto que participa da construção.
    - Cria um gráfico de tarefas para as tarefas solicitadas.
- Execução
    - Agenda e executa cada uma das tarefas selecionadas na ordem de suas dependências.

Vale detalhar que na fase de inicialização, o Gradle detecta o conjunto de projetos e compilações incluídas que participam da compilação. Gradle primeiro avalia o arquivo de configurações. Em seguida, Gradle instancia *Project* instâncias para cada projeto.

Importante a detecção do arquivo de configurações quando o Gradle é executado, pois o há um diretório que contém um arquivo chamado *settings.gradle*, o Gradle usa o *settings.gradle* para inicializar a compilação. O Gradle pode ser executado em qualquer subprojeto da compilação. Ao executar o Gradle em um diretório que não contém nenhum *settings.gradle*, os seguintes passos serão executados:

1 – Gradle procura um *settings.gradle* (.kts) nos diretórios pais.

2 – Se o Gradle encontrar um *settings.gradle* (.kts), o Gradle verificará se o projeto atual faz parte da compilação de vários projetos. Nesse caso, o Gradle é construído como um projeto múltiplo.

3 – Se o Gradle não encontrar um *settings.gradle* (.kts), o Gradle cria como um único projeto.

Por ora, observa-se que os gerenciadores de dependências fazem parte da cultura e ciclo de vida DevOps[3] e são importantes para Java e o padrão de desenvolvimento atual.

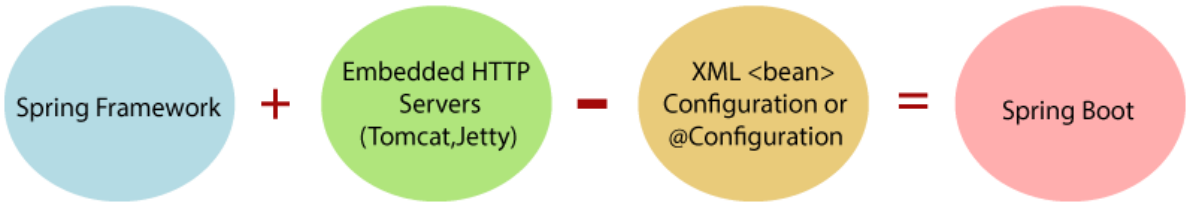
[2] Essa interface é a API principal que você usa para interagir com o Gradle a partir do seu arquivo de compilação. Em um Project, você tem acesso programático a todos os recursos do Gradle. https://docs.gradle.org/current/dsl/org.gradle.api.Project.html

[3] O ciclo de vida do DevOps são os vários estágios que envolvem o ecossistema DevOps. O ecossistema DevOps é o ambiente com as ferramentas e tecnologias certas implementadas para praticar a cultura DevOps no desenvolvimento de software. Temos um grande número de ferramentas e tecnologias que estão implementando

2. Spring Boot

O Spring Boot é um projeto construído sobre o Spring Framework[4]. Ele fornece uma maneira mais fácil e rápida de instalar, configurar e executar aplicativos simples e baseados na web. O Java Spring Boot é uma ferramenta de código aberto que facilita o uso de estruturas baseadas em Java para criar microsserviços[5] e aplicativos da web. Para qualquer definição de Spring Boot, a conversa deve começar com Java – uma das linguagens de desenvolvimento e plataformas de computação mais populares e amplamente utilizadas para desenvolvimento de aplicativos. Desenvolvedores de todo o mundo iniciam sua jornada de codificação aprendendo Java. Flexível e fácil de usar, o Java é o favorito do desenvolvedor para uma variedade de aplicativos - tudo, desde aplicativos de mídia social, web e jogos até aplicativos empresariais e de rede.

Pode-se dizer que o Spring Boot é um módulo Spring que fornece o recurso de desenvolvimento rápido de aplicações ou Rapid Application Development(RAD) para o Spring Framework. Ele é usado para criar um aplicativo autônomo baseado em Spring que você pode simplesmente executar porque precisa de uma configuração Spring mínima ou seja é a combinação de Spring Framework e Embedded Servers:



No Spring Boot, não há exigência de configuração XML (descritor de implantação). Ele usa a convenção sobre o paradigma de design de software de configuração, o que significa que diminui o esforço do desenvolvedor.

---

a prática DevOps no mercado e identificar todas elas é muito difícil. Neste artigo, abordaremos sobre as diversas ferramentas e tecnologias que estão envolvidas nas diversas etapas do ecossistema DevOps.

[4] Spring Framework é uma plataforma Java que fornece suporte de infraestrutura abrangente para o desenvolvimento de aplicativos Java. O Spring lida com a infraestrutura para que você possa se concentrar em seu aplicativo.
O Spring permite que você crie aplicativos a partir de "objetos Java simples" (POJOs) e aplique serviços corporativos de forma não invasiva a POJOs. Esse recurso se aplica ao modelo de programação Java SE e ao Java EE completo e parcial.

[5] Microsserviços são uma abordagem arquitetônica e organizacional do desenvolvimento de software na qual o software consiste em pequenos serviços independentes que se comunicam usando APIs bem definidas. Esses serviços pertencem a pequenas equipes autossuficientes.

Importante entender que se deve utilizar o Spring Boot Framework porque:

- A abordagem de injeção de dependência é usada no Spring Boot.
- Contém recursos poderosos de gerenciamento de transações de banco de dados.
- Simplifica a integração com outros frameworks Java como JPA/Hibernate ORM, Struts, etc.
- Reduz o custo e o tempo de desenvolvimento do aplicativo.
- Cria aplicativos Spring independentes que podem ser iniciados usando Java -jar.
- Testa aplicativos da web facilmente com a ajuda de diferentes servidores HTTP incorporados, como Tomcat, Jetty, etc. Não precisamos implantar arquivos WAR.
- Fornece POMs 'iniciais' opinativos para simplificar nossa configuração Maven.
- Fornece recursos prontos para produção, como métricas, verificações de integridade e configuração externa.
- Não há nenhum requisito para configuração de XML.
- Oferece uma ferramenta CLI para desenvolver e testar o aplicativo Spring Boot.
- Oferece o número de plug-ins.
- Também minimiza a gravação de vários códigos padrão (o código que deve ser incluído em muitos lugares com pouca ou nenhuma alteração), configuração XML e anotações.
- Aumenta a produtividade e reduz o tempo de desenvolvimento.

Vale ressaltar que uma desvantagem ou limitação do Spring Boot é a utilização de dependências que não serão usadas no aplicativo, aumentando o tamanho do aplicativo/software.

Para facilitar a utilização do Spring Boot uma ferramenta chamada de Spring initializr foi disponibilizada, por meio de uma interface web bem simples, permitindo a geração de um projeto a partir de uma estrutura de configurações pré-moldadas.

Possibilita a realização de configurações de versões do java/spring boot, grupo/nome do projeto, série de lista de dependências e etc.

Com poucos passos o projeto é gerado, conforme descreve-se:

1. Acessar a interface web do Spring Initializr no endereço https://start.spring.io/
2. Escolher a linguagem – Java ou Kotlin ou Groovy
3. Selecionar a versão
4. Preencher os metadados
    a. Group — geralmente o domínio reverso da empresa ou organização;
    b. Artifact — o artefato a ser gerado (nome da aplicação);
    c. Name — será automaticamente preenchido com o mesmo valor do campo Artifact;
    d. Description — uma rápida descrição do projeto;
    e. Package name — estrutura do pacote inicial da aplicação;
    f. Packaging — como será empacotada a aplicação, no caso da disciplina o padrão Jar;
    g. Java — versão do java utilizada.
5. Adicionar as dependências do projeto.
6. Gerar o projeto clicando no botão Generate. O artefato do projeto será baixado em formato zip.
7. Importar o projeto para IDE.

Ainda dentro do contexto existe Spring Boot Starters que conceitua e implementa a descrição de dependências.

Vale ressaltar que não se utiliza spring-boot-starter para incluir dependências de terceiros, pois é reservado para artefatos oficiais Spring Boot, como os da tabela abaixo:

Nome	Descrição
spring-boot-starter-thymeleaf	Ele é usado para criar aplicativos da web MVC usando visualizações Thymeleaf.
spring-boot-starter-data-couchbase	Ele é usado para o banco de dados orientado a documentos Couchbase e Spring Data Couchbase.

spring-boot-starter-artemis	Ele é usado para mensagens JMS usando o Apache Artemis.
spring-boot-starter-web-services	É usado para Spring Web Services.
spring-boot-starter-mail	Ele é usado para suportar o envio de e-mail do Java Mail e do Spring Framework.
spring-boot-starter-data-redis	Ele é usado para armazenamento de dados de valor-chave Redis com Spring Data Redis e o cliente Jedis.
spring-boot-starter-web	Ele é usado para construir o aplicativo da web, incluindo aplicativos RESTful usando Spring MVC. Ele usa o Tomcat como o contêiner incorporado padrão.
spring-boot-starter-data-gemfire	Ele é usado para armazenar dados distribuídos GemFire e Spring Data GemFire.
spring-boot-starter-activemq	Ele é usado em mensagens JMS usando o Apache
spring-boot-starter-data-elasticsearch	Ele é usado no mecanismo de pesquisa e análise do Elasticsearch e no Spring Data Elasticsearch.
spring-boot-starter-integration	Ele é usado para integração do Spring.
spring-boot-starter-test	Ele é usado para testar aplicativos Spring Boot com bibliotecas, incluindo JUnit, Hamcrest e Mockito.
spring-boot-starter-jdbc	Ele é usado para JDBC com o conjunto de conexões Tomcat JDBC.
spring-boot-starter-mobile	Ele é usado para criar aplicativos da Web usando o Spring Mobile.
spring-boot-starter-validation	Ele é usado para validação de Java Bean com Hibernate Validator.
spring-boot-starter-hateoas	Ele é usado para construir um aplicativo da Web RESTful baseado em hipermídia com Spring MVC e Spring HATEOAS.
spring-boot-starter-jersey	Ele é usado para construir aplicativos da Web RESTful usando JAX-RS e Jersey. Uma alternativa ao spring-boot-starter-web.
spring-boot-starter-data-neo4j	Ele é usado para o banco de dados de gráficos Neo4j e Spring Data Neo4j.
spring-boot-starter-data-ldap	Ele é usado para Spring Data LDAP.
spring-boot-starter-websocket	Ele é usado para construir os aplicativos WebSocket. Ele usa o suporte WebSocket do Spring Framework.
spring-boot-starter-aop	É usado para programação orientada a aspectos com Spring AOP e AspectJ.
spring-boot-starter-amqp	É usado para Spring AMQP e Rabbit MQ.
spring-boot-starter-data-cassandra	É usado para banco de dados distribuído Cassandra e Spring Data Cassandra.
spring-boot-starter-social-facebook	É usado para o Spring Social Facebook.
spring-boot-starter-jta-atomikos	Ele é usado para transações JTA usando Atomikos.

spring-boot-starter-security	É usado para Spring Security.
spring-boot-starter-bigode	Ele é usado para criar aplicativos da Web MVC usando exibições Mustache.
spring-boot-starter-data-jpa	É usado para Spring Data JPA com Hibernate.
spring-boot-starter	Ele é usado como iniciador principal, incluindo suporte de configuração automática, criação de log e YAML.
spring-boot-starter-groovy-templates	Ele é usado para criar aplicativos da Web MVC usando visualizações de modelo Groovy.
spring-boot-starter-freemarker	Ele é usado para criar aplicativos da Web MVC usando exibições do FreeMarker.
spring-boot-starter-lote	É usado para Spring Batch.
spring-boot-starter-social-linkedin	É usado para o Spring Social LinkedIn.
spring-boot-starter-cache	Ele é usado para o suporte de cache do Spring Framework.
spring-boot-starter-data-solr	Ele é usado para a plataforma de pesquisa Apache Solr com Spring Data Solr.
spring-boot-starter-data-mongodb	Ele é usado para banco de dados orientado a documentos MongoDB e Spring Data MongoDB.
spring-boot-starter-jooq	Ele é usado para o jOOQ acessar bancos de dados SQL. Uma alternativa para spring-boot-starter-data-jpa ou spring-boot-starter-jdbc.
spring-boot-starter-jta-narayana	Ele é usado para o Spring Boot Narayana JTA Starter.
spring-boot-starter-cloud-conectores	É usado para Spring Cloud Connectors que simplifica a conexão a serviços em plataformas de nuvem como Cloud Foundry e Heroku.
spring-boot-starter-jta-bitronix	É usado para transações JTA usando Bitronix.
spring-boot-starter-social-twitter	É usado para o Spring Social Twitter.
spring-boot-starter-data-rest	Ele é usado para expor repositórios Spring Data sobre REST usando Spring Data REST.

Por ora, resta mencionar que após as configurações e implementações do Spring boot contribuíram para a celeridade e organização de um projeto de aplicativo/software.

Referências

- ★ https://www.baeldung.com/maven
- ★ https://maven.apache.org/guides/introduction/introduction-to-the-pom.html
- ★ https://ant.apache.org/
- ★ https://docs.gradle.org/current/userguide/publishing_ivy.html
- ★ https://acervolima.com/diferenca-entre-maven-e-ant/
- ★ https://docs.gradle.org/current/userguide/what_is_gradle.html
- ★ https://www.tutorialspoint.com/apache_ivy/apache_ivy_overview.htm
- ★ https://ant.apache.org/ivy/history/2.2.0/use/publish.html
- ★ https://docs.gradle.org/current/userguide/build_lifecycle.html
- ★ https://docs.gradle.org/current/dsl/org.gradle.api.Project.html
- ★ https://digitalvarys.com/tools-for-devops/

★ https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-java-spring-boot/
★ https://www.javatpoint.com/spring-boot-tutorial
★ https://start.spring.io/