Primeiros passos com Spring BOOT

O Spring Boot é uma ferramenta que visa facilitar o processo de configuração e publicação de aplicações que utilizem o ecossistema Spring.

O Spring Boot fornece a maioria dos componentes baseados no Spring, necessários em aplicações de maneira pré-configurada, tornando possível termos uma aplicação rodando em produção rapidamente e com o esforço mínimo de configuração e implantação.

 **Documentação do Spring Boot**

Para facilitar ainda mais, o Spring disponibiliza a página **Spring Initializr**. Nesta página, com alguns poucos cliques, você pode criar um projeto inteiro.

No final, a página irá gerar um projeto **Maven** ou **Gradle**, que são gerenciadores de dependências da linguagem Java (semelhante ao NPM do JavaScript/Typescript), pré-configurado e com todos os componentes solicitados especificados. Nossos projetos Spring Boot utilizarão o **Maven** como Gerenciador Dependências.

1. Apache Maven

A palavra Maven significa acumulador de conhecimento. No Universo Java, o Maven é uma ferramenta usada para construir e gerenciar qualquer projeto Java, tornando o trabalho diário dos desenvolvedores mais fácil, além de simplificar a compreensão de qualquer projeto baseado na linguagem Java.

 **Documentação: Maven**

Entre as principais características do Maven, destaca-se:

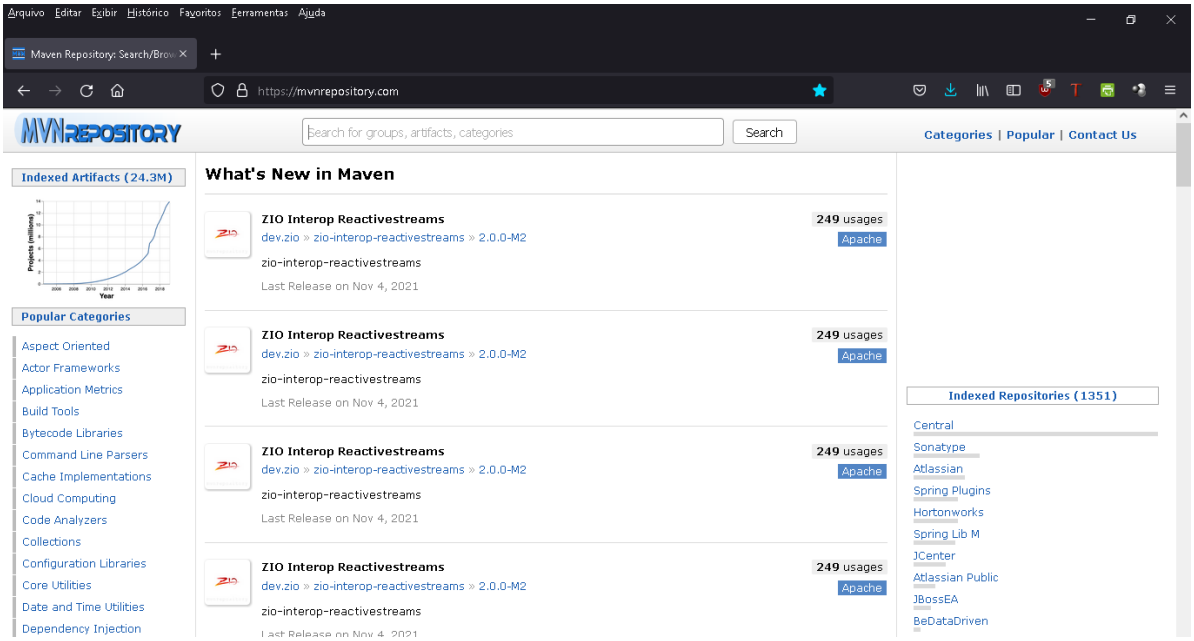
1.1 Gerenciador de dependências

O Maven é responsável por fazer o download das bibliotecas que você vai precisar no seu projeto. Para efetuar esta tarefa, o Maven utiliza o arquivo pom.xml, onde você precisa declarar todas as dependências necessárias para o seu projeto.

1.2 Repositório central

Todas as ferramentas e bibliotecas utilizadas nos projetos Spring Boot estão disponíveis em um único servidor na nuvem chamado **Maven Central Repository**. O Maven Repository facilita e centraliza o download de todas as dependências independente de serem as oficiais do Spring ou Desenvolvidas por outras Empresas ou Pessoas Desenvolvedoras (Lombok, Flyway, MOckito, entre outras), dispensando a necessidade de procurar as dependências no Google, por exemplo.

 **Site Oficial: Maven Repository**



1.3 Automatizador de tarefas

Um projeto que possui muitas bibliotecas e muitas dependências gera alguns problemas no dia a dia, tais como: manter todas atualizadas, fazer o build da sua aplicação, realizar alguns testes e etc. O **MAVEN** auxilia nestes e outros processos através dos seus scripts prontos que automatizam todas estas tarefas.

DICA: ***Além do Maven, existe um outro Gerenciador de Dependências para projetos Spring Boot chamado Gradle. Para conhecer o Gradle, acesse: https://gradle.org/***

2. Como funciona um projeto SPRING BOOT?

1. A Classe Principal, que possui o método main, inicia um servidor WEB (TOMCAT), que vai gerenciar todas as URL's (Endpoints) disponíveis na API.
2. Cada URL deve ser mapeada para um determinado método de uma classe.
3. A execução desse método retornará uma resposta quando acionamos a URL.
4. A partir daí, criamos nossos objetos que implementarão todas as lógicas necessárias.

 **Site Oficial: Servidor WEB TOMCAT**

2.1 Como planejar um projeto SPRING BOOT?

Quais ENDPOINTS vamos oferecer? (Um Endpoint é uma URL associada a um método do protocolo HTTP: GET, POST, PUT, DELETE).

Em geral, temos 1 Endpoint para cada método HTTP (podemos ter mais de um desde que os endereços sejam diferentes), em cada objeto do nosso modelo de negócios:

Objeto de Negócios: **PRODUTO**

- URL para recuperar dados de um produto (GET)
- URL para inserir novo produto (POST)
- URL para atualizar dados de um produto (PUT)

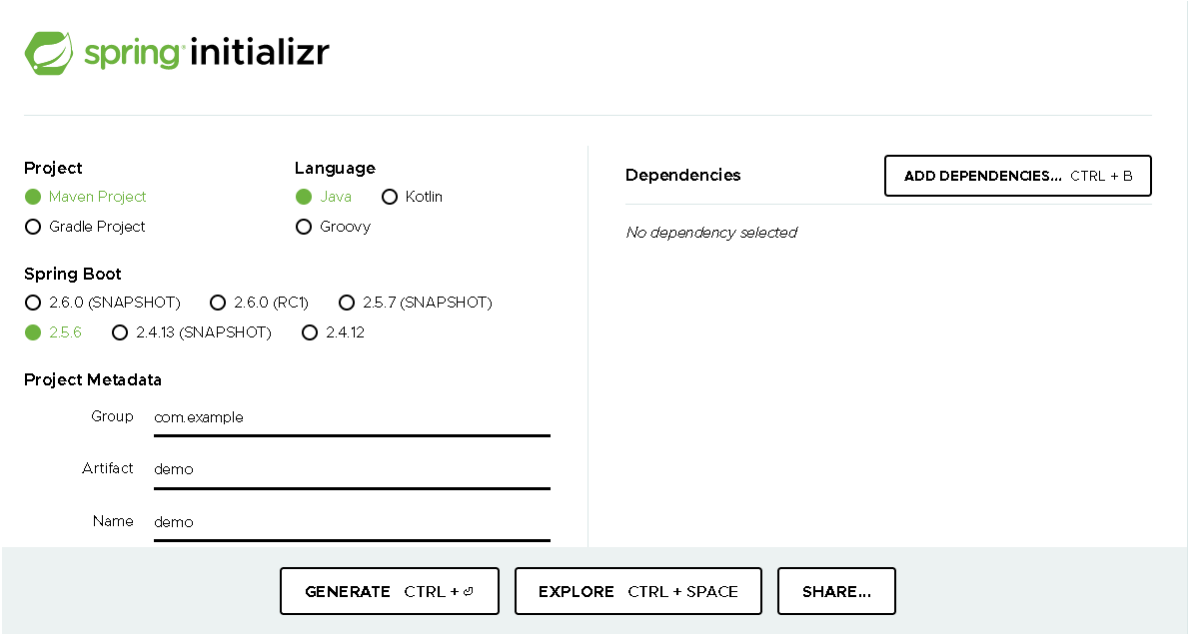
- URL para remover um produto do sistema (DELETE)

Como estes ENDPOINTS estarão estruturados no nosso sistema? Vamos discutir isto Mais adiante...

Projeto 01 - Hello World!

Passo 01 - Abrindo o Spring Initializr

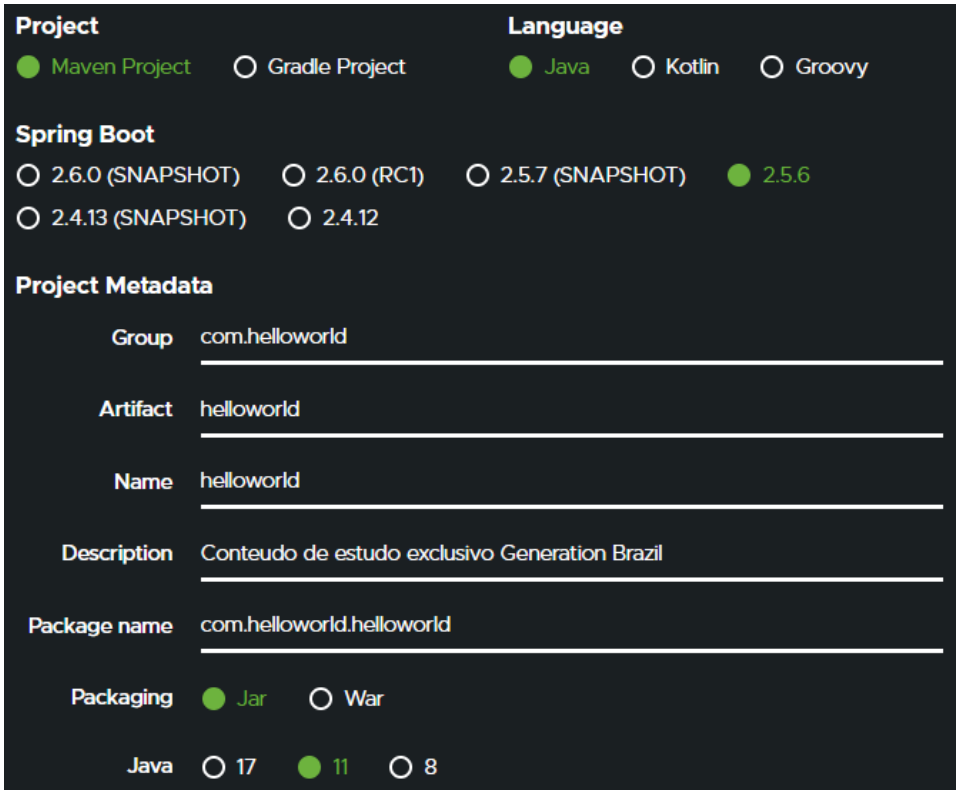
1. Abra o Navegador de sua preferência e acesse o **Spring Initializr**, através do endereço: https://start.spring.io



Passo 02 - Setup do Projeto

2.1 Configurações iniciais

Vamos configurar o template inicial do nosso projeto conforme a figura abaixo:

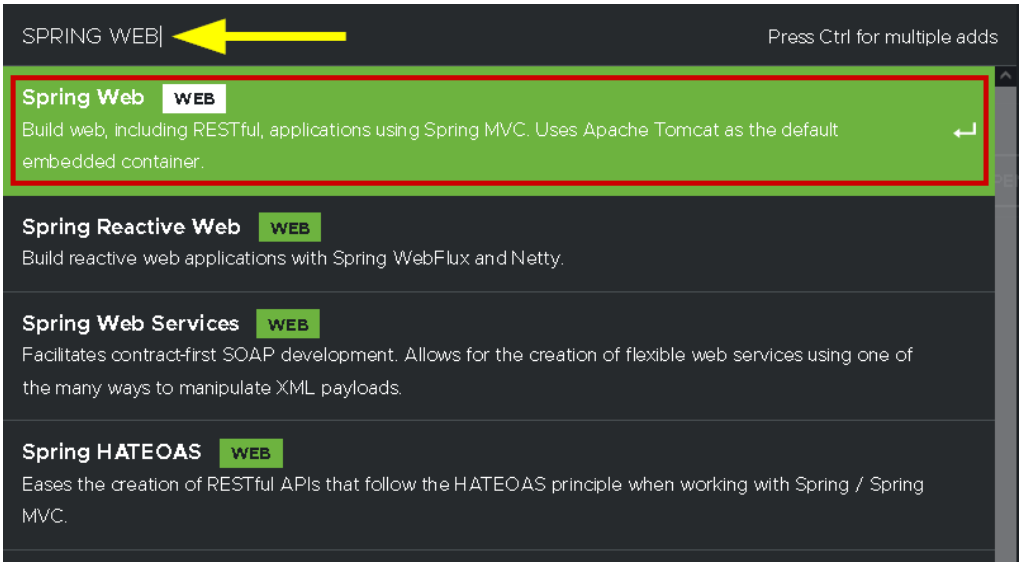


Item	Descrição
Project	Define o Gerenciador de Dependências (Maven Project).
Language	Define a Linguagem (Java).
Spring Boot	Define a versão do Spring Boot, que será utilizada. Mantenha a versão indicada pelo Spring Initializr.
Group	O Endereço reverso do Domínio da sua Empresa ou Organização. **Exemplo:** *generation.com* → ***com.generation***
Artifact	O artefato a ser gerado, ou seja, o mesmo nome do projeto.
Name	Nome do Projeto (letras minúsculas, sem acentos ou espaços).
Description	Breve descrição do projeto.
Package Name	Estrutura do pacote inicial da aplicação (Group + Artifact). **Exemplo:** ***com.generation.helloworld***
Packaging	Define como a aplicação será empacotada (JAR).
Java Version	Versão do Java (a versão da imagem pode ser diferente da sua tela).

2.2 Dependências

No projeto **Hello World!** vamos adicionar 2 dependências: **Spring Web** e **Spring Boot Dev Tools**.

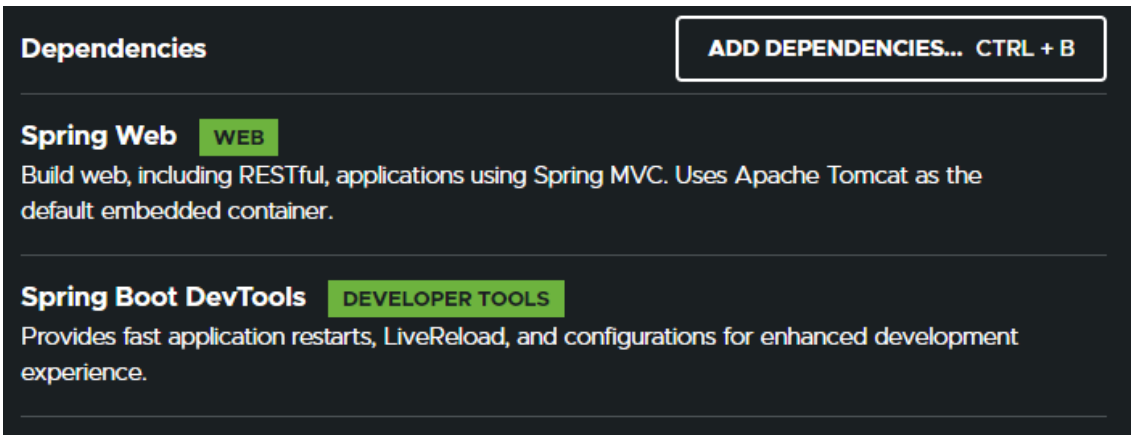
1. Na página do Spring Initializr, cique no botão **ADD DEPENDENCIES... CTRL + B**
2. Na caixa de pesquisa (indicada na figura com uma seta amarela), digite o nome da dependência que você deseja adicionar e clique sobre o nome da Dependência para adicionar (indicada na figura com um retângulo vermelho)



3. Repita os itens 2 e 3 para adicionar a segunda Dependência (Spring Boot Dev Tools)

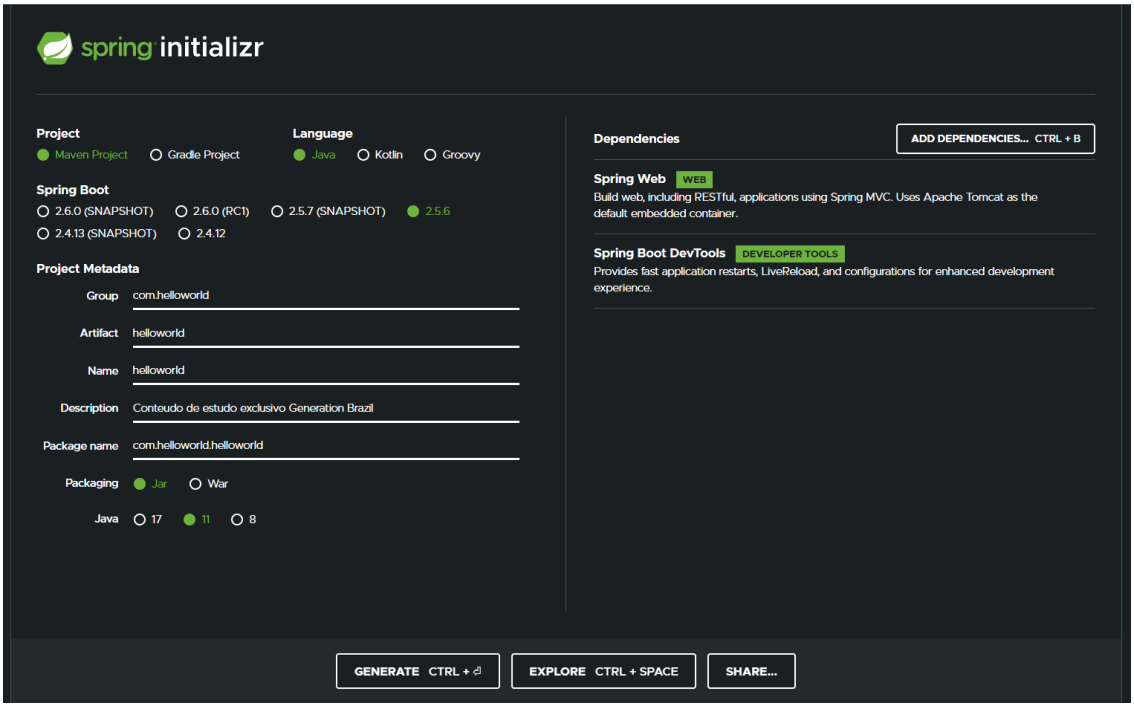
> **DICA:** ***Mantenha a tecla CTRL pressionada ao clicar sobre a Dependência que será adicionada ao projeto. Desta forma, a janela se manterá aberta e você conseguirá selecionar todas Dependências do projeto de uma só vez.***

4. Após adicionar as duas Dependências, o item Dependencies do Spring Intializr estará igual a figura abaixo:



Dependência	Descrição
Spring Web	Fornece todas as Bibliotecas necessárias para trabalhar com o protocolo HTTP.
Spring Boot Dev Tools	Permite a atualização do projeto em tempo real durante o processo de Desenvolvimento da aplicação.

5. O seu projeto deverá estar semelhante a figura abaixo:



6. Clique em **Generate Ctrl +** ↵ (CTRL + Enter)
7. Faça o Download do Projeto

🦶 Passo 03 - Importando o Projeto no STS

Para desenvolver os nossos projetos Spring utilizaremos a IDE **Spring Tools Suite (STS)**. O STS é baseado no **Eclipse** e já está totalmente configurada para executar o Spring Framework, sem a necessidade de instalar outros aplicativos.
A única exigência do STS é que você possua o **Java Development Kit - JDK** instalado no seu computador (mínimo versão 11).

[Site Oficial: STS]

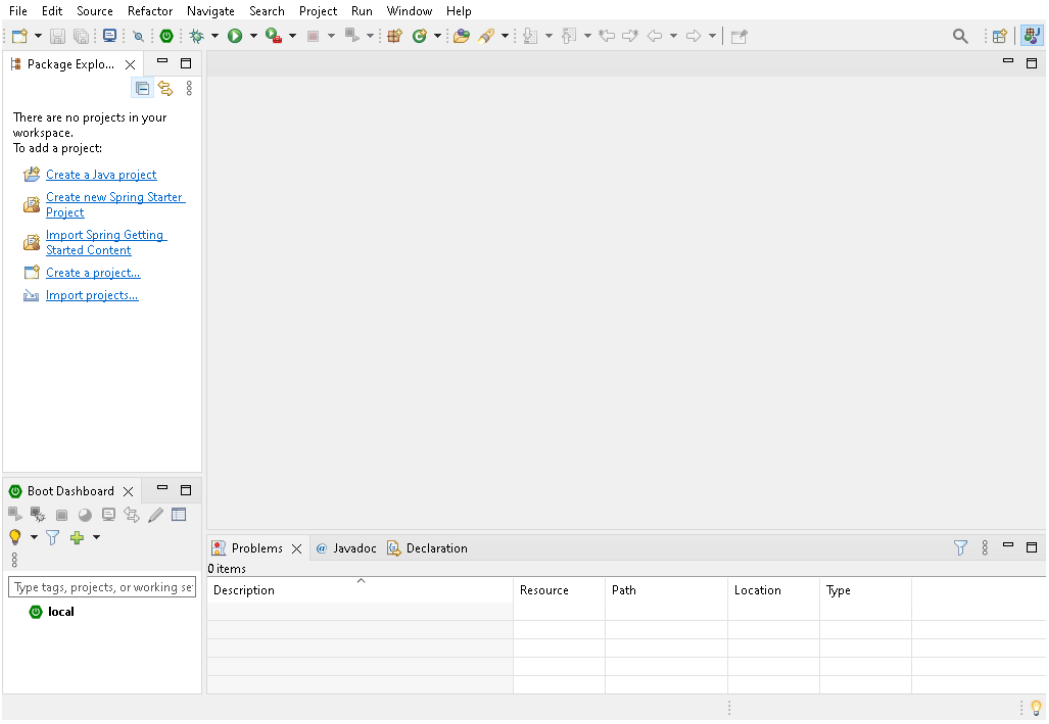
[Site Oficial: Java Development Kit]

💡	**DICA:** ***Caso você tenha alguma dúvida quanto a instalação do STS, consulte o Guia de Instalação do STS.***

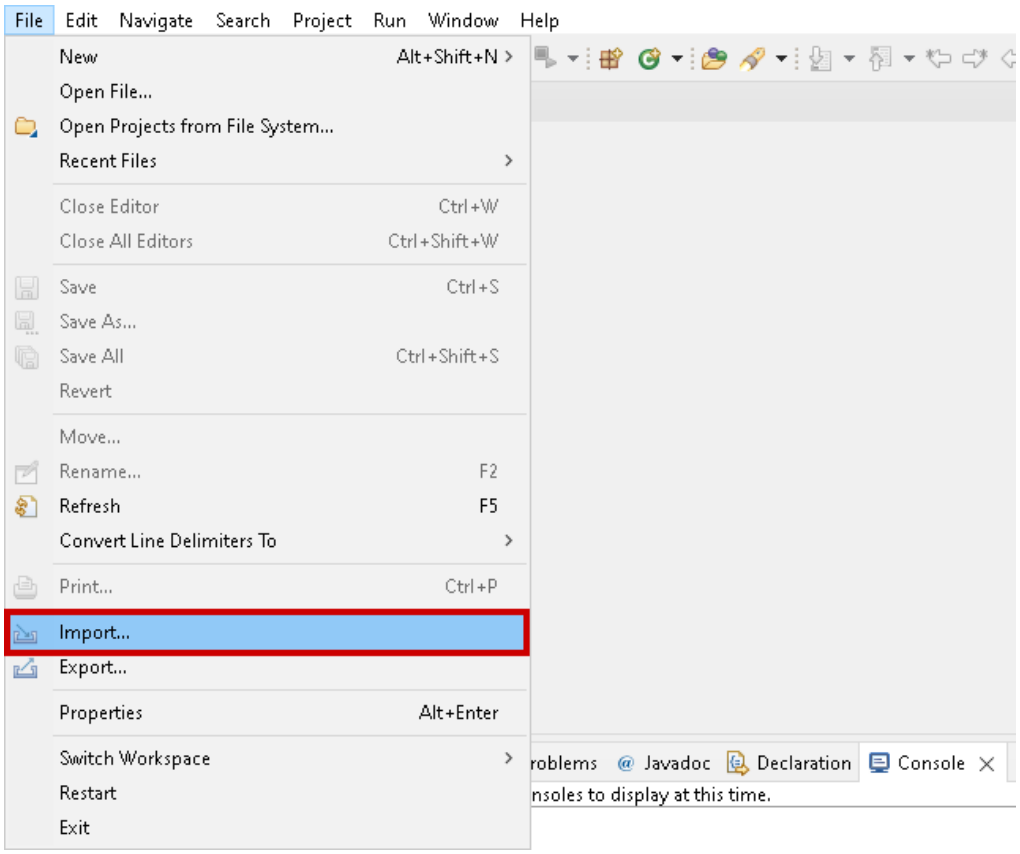
1. Extraia o arquivo zip dentro da pasta Workspace do STS

❗	**ATENÇÃO:** ***A pasta Workspace do STS geralmente está localizada em: C:\Users\seu usuario\Documents\workspace-spring-tool-suite-4-4.12.1.RELEASE***

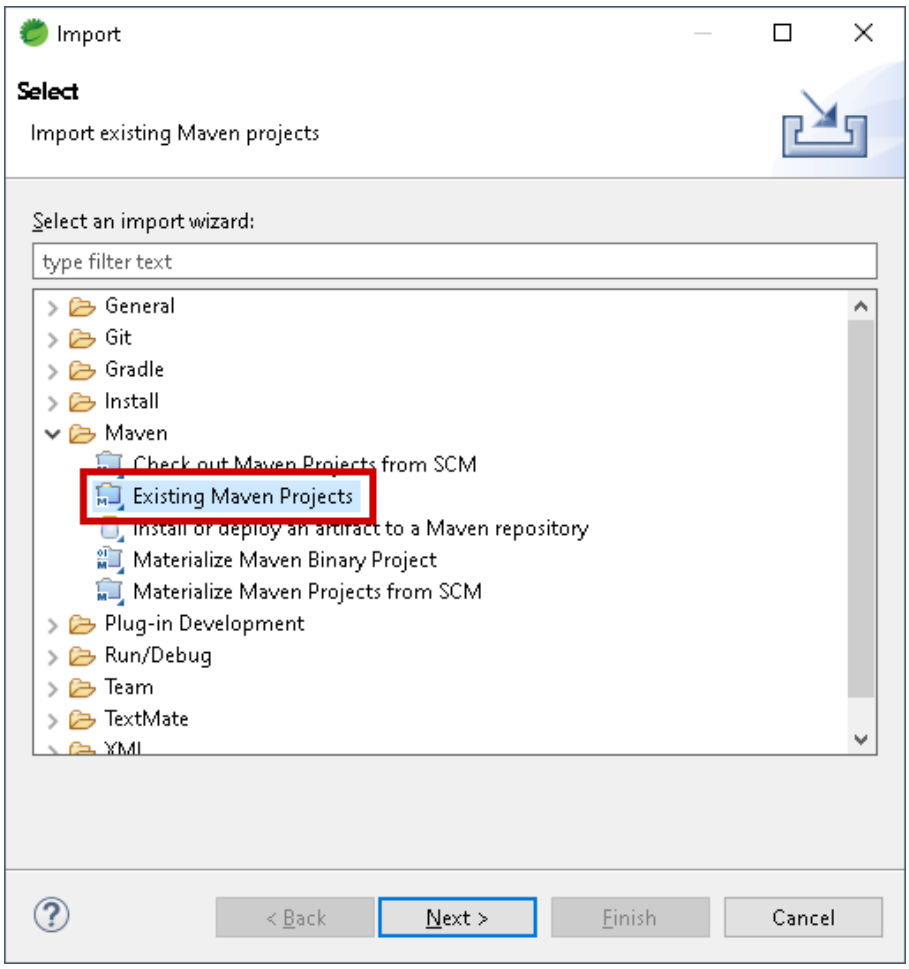
2. Abra o STS



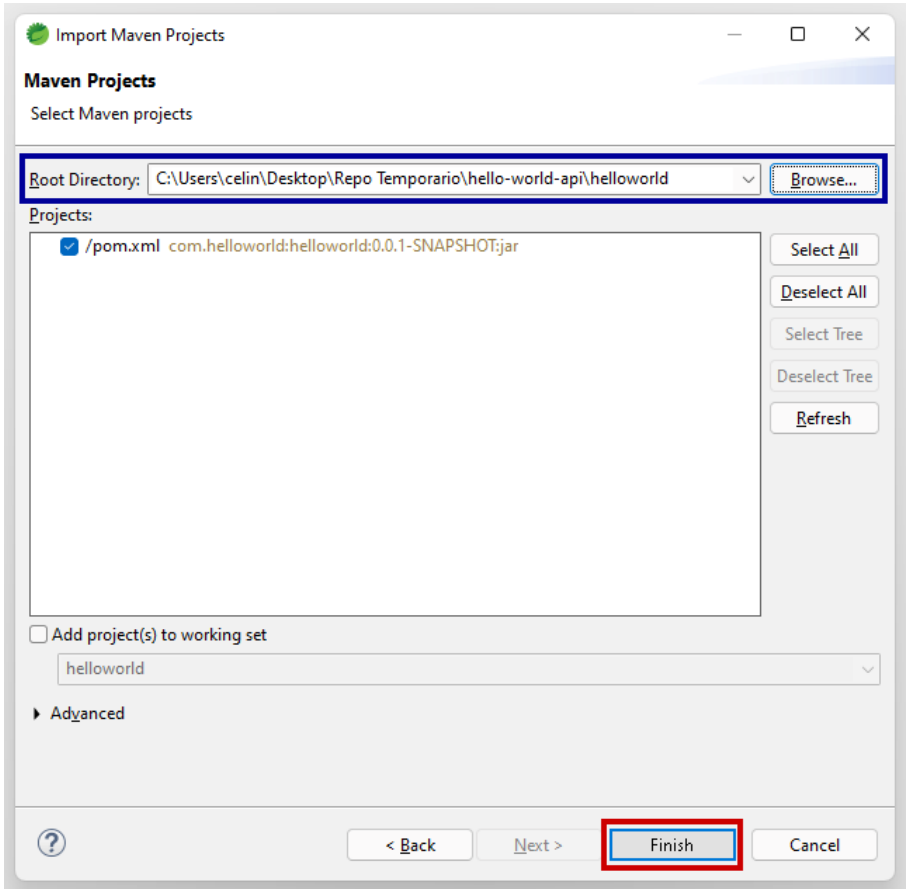
3. Importe o projeto no STS através da opção **File → Import**



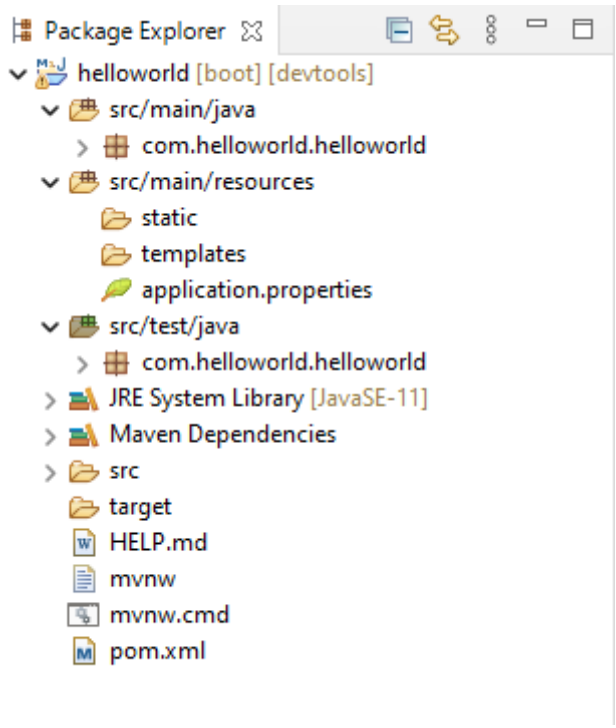
4. Na janela **Import**, clique na pasta **Maven**, na opção **Existing Maven Projects**



5. No item **Root Directory**, clique no botão **Browse** e selecione a pasta do projeto (indicado em azul na imagem abaixo).
6. No item **Projects**, selecione o arquivo **pom.xml** e clique no botão **Finish** para concluir a importação



7. A estrutura do nosso projeto importado para o STS ficará de acordo com a imagem abaixo:



- As pastas cujo nome iniciam com **src** são chamadas de **Source Folder**.
- As estruturas que estão contidas dentro das pastas **src/main/java** e **src/test/java** são chamadas de **Package** (Pacote).
- Na pasta **src/main/java** está localizado o pacote principal da aplicação. Todas as Camadas da aplicação são desenvolvidas dentro deste pacote.
- Na pasta **src/test/java** está localizado o pacote de testes da aplicação. Todas as Camadas de teste da aplicação são desenvolvidas dentro deste pacote.

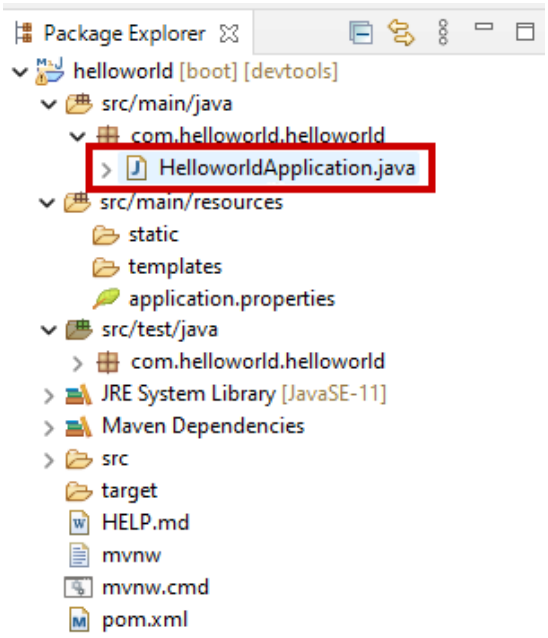
3.1 Entendendo a Estrutura do nosso projeto

Item	Descrição
src/main/java	Source Folder mais importante da aplicação, onde será desenvolvido todo o código da nossa aplicação dentro do pacote principal (em nosso exemplo: **com.helloworld.helloworld**). Dentro deste pacote existe um arquivo com o **nome do projeto + a palavra Application** (em nosso exemplo: **HelloworldApplication.java**), que é responsável por inicializar a aplicação (Classe Main). **Não apague este arquivo ou altere a estrutura de pastas do projeto**.
src/main/resources	Source Folder responsável por manter os recursos da aplicação, ou seja, os arquivos de configuração do projeto. O mais importante deles é o **application.properties**, que é o responsável por manter as configurações de Data, Hora, Fuso-horário, Banco de Dados, entre outras.
src/test/java	Source Folder onde serão desenvolvidas as Classes de Teste da aplicação, dentro do pacote de testes. Observe que o nome do pacote é o mesmo do pacote da Source Folder Principal.
JRE System Library	Neste pacote, o Maven faz o download do Compilador Java (JDK) durante a importação do projeto para o STS
Maven Dependencies	Neste pacote, o Maven faz o download de todas as Dependências inseridas no projeto (ver arquivo pom.xml) durante a importação do projeto para o STS
pom.xml	O **Project Object Model (POM)** é o arquivo principal de configuração do **Maven**. É um arquivo XML que contém informações sobre o projeto e detalhes de configuração usados pelo Maven para construir o projeto. **Não apague este arquivo e ao fazer alterações tenha muito cuidado para manter a estrutura do arquivo**.

🦶 Passo 04 - Conhecendo a Classe Principal do projeto

A Classe principal do projeto (Main), é responsável por inicializar o nosso projeto. Esta classe é gerada automaticamente pelo Spring Boot durante a criação do projeto e raramente precisaremos fazer alterações nela. Vamos abrir a Classe **HelloworldApplication.java** (Classe principal do nosso projeto) e conhecer o seu conteúdo.

1. No **STS**, na **Package Explorer**, clique na pasta **src/main/java** e na sequência clique no pacote principal **com.helloworld.helloworld**.
2. Clique duas vezes sobre o arquivo **HelloworldApplication.java**.



3. Será aberto o código abaixo:

```
1  package com.helloworld.helloworld;
2
3  import org.springframework.boot.SpringApplication;
5
6  @SpringBootApplication
7  public class HelloworldApplication {
8
9      public static void main(String[] args) {
10         SpringApplication.run(HelloworldApplication.class, args);
11     }
12
13 }
14
```

Na **Linha 06** foi invocada a anotação **@SpringBootApplication** que indica que o nosso projeto é do tipo Spring Boot Application. Esta anotação inicializa uma série de configurações padrão em projetos Spring Boot, além de scannear os pacotes em busca das Classes que compõe a aplicação, inicializando as configurações especificas de cada uma.

Na **Linha 10** o método **run** da Classe **SpringApplication** inicializa a aplicação executando a Classe Principal da aplicação (em nosso exemplo: **HelloworldApplication.java**). A classe SpringApplication fornece uma maneira simples de inicializar um aplicativo Spring que é iniciado a partir de um método main() (Semelhante ao do Java), onde é indicada a Classe principal da Aplicação.

Documentação: *@SpringBootApplication*

Documentação: *@SpringApplication* e o *Método run*

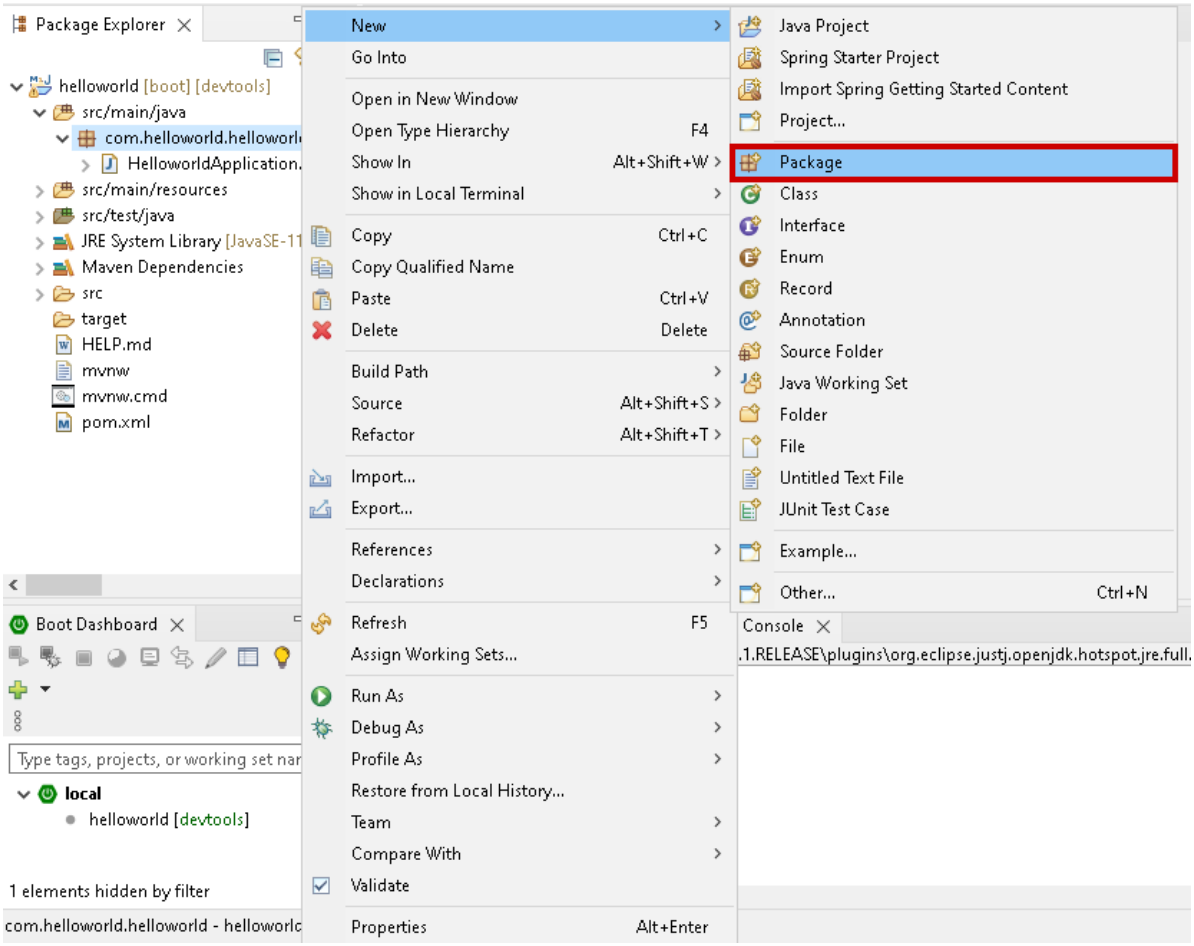
Código fonte da Classe Main

Passo 05 - Criando a primeira Classe Controller

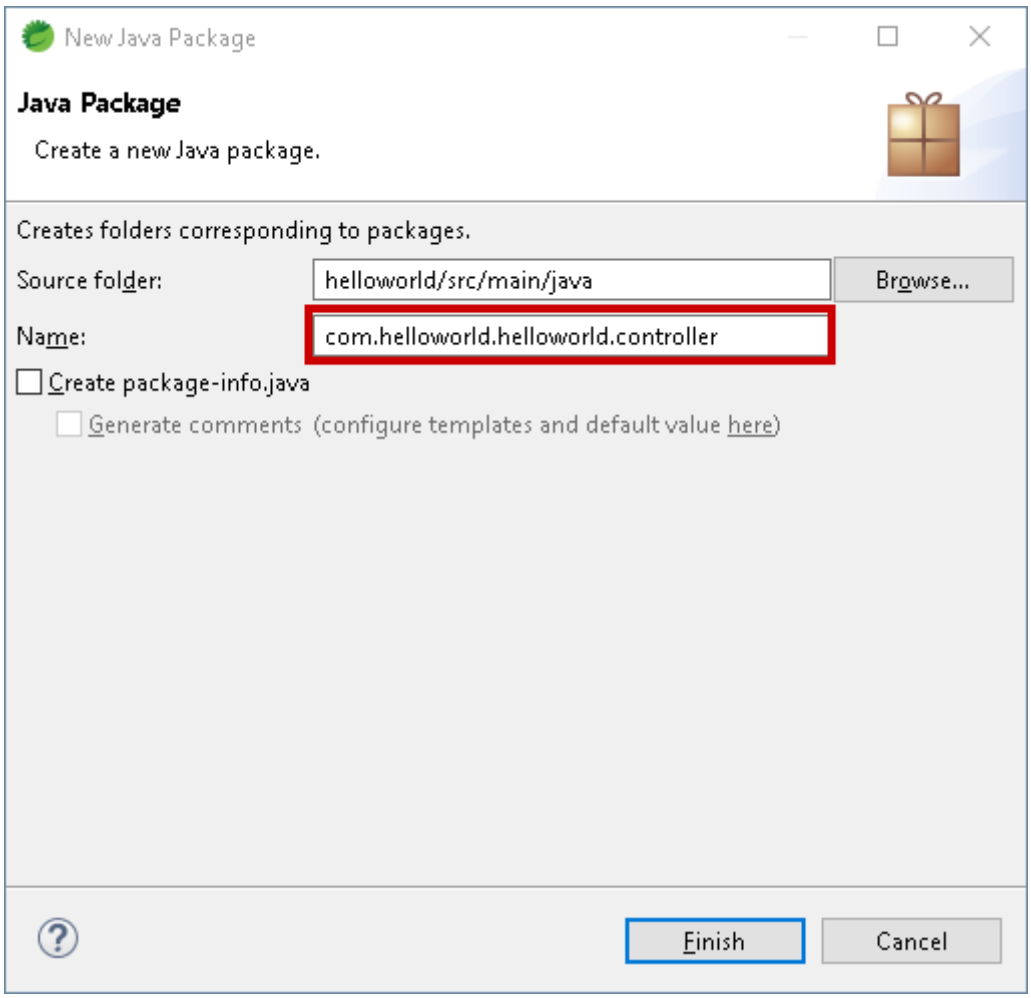
ALERTA DE BSM: ***Mantenha a Atenção aos Detalhes ao criar as Classes do projeto. As Classes devem ser criadas no Pacote Main na Source Folder src/main/java. Crie as Classes com calma e não altere a estrutura de pastas do projeto.***

Primeiro vamos criar o pacote **Controller**, onde será criada a nossa Classe controladora.

1. Clique com o botão direito do mouse sobre o pacote principal da aplicação (em nosso exemplo: **com.helloworld.helloworld**).
2. Na sequência, clique na opção **New → Package**



3. Na janela **New Java Package**, no item **Name**, acrescente no final do nome do pacote **.controller**, como mostra a figura abaixo:

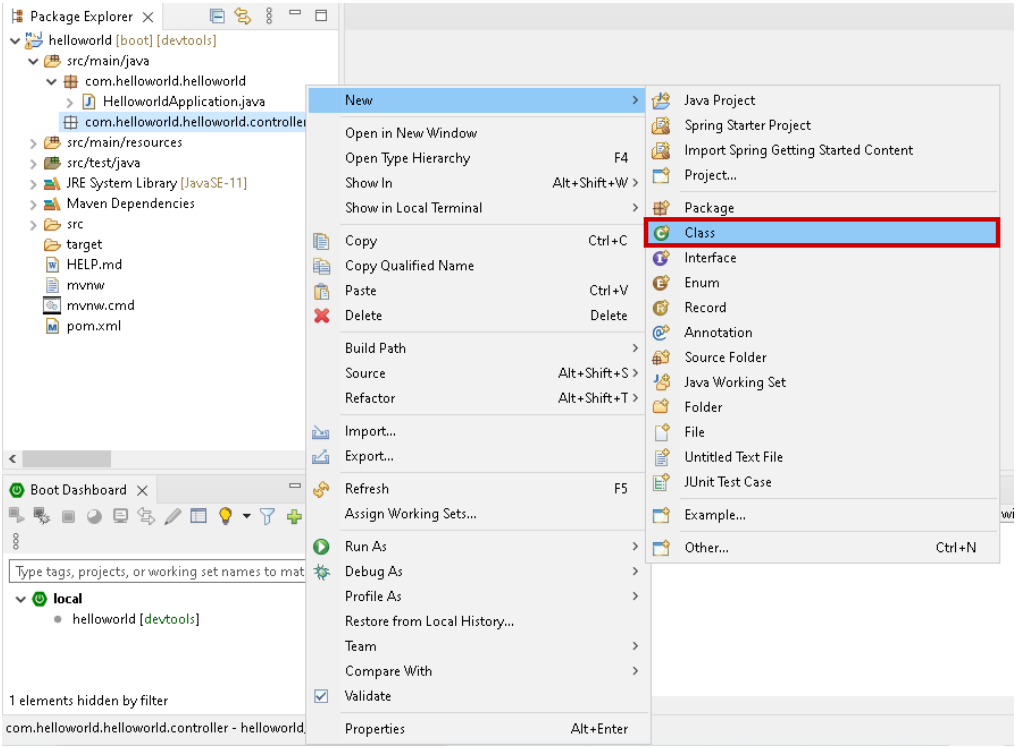


4. Clique no botão **Finish** para concluir.

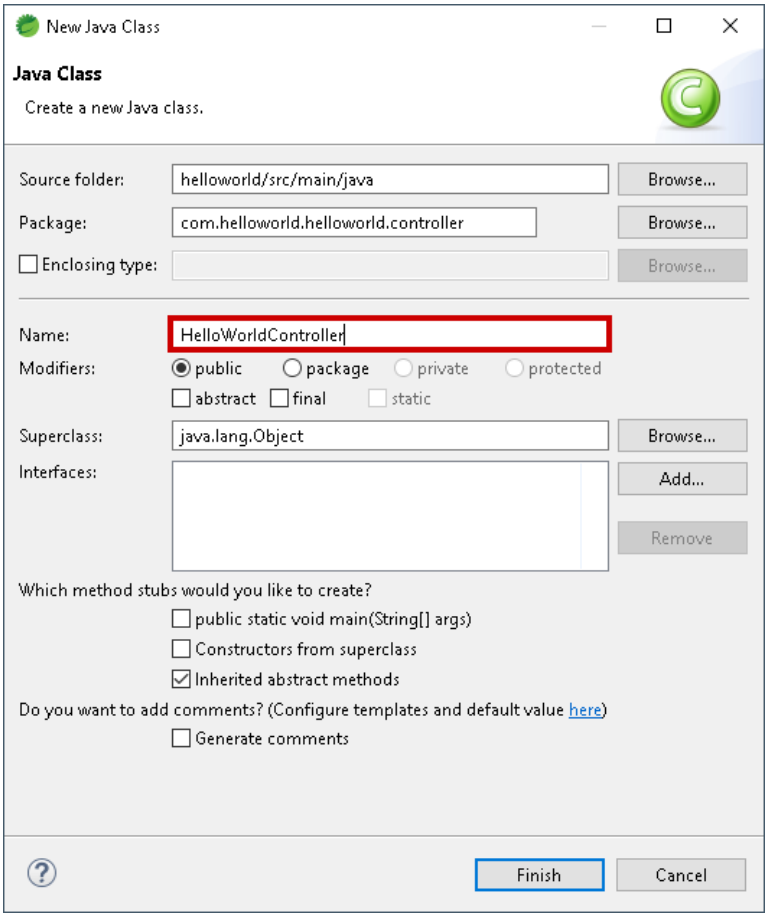
Na sequência, vamos criar a Classe Controladora que chamaremos de **HelloWorldController**.

1. Clique com o botão direito do mouse sobre o pacote controller da aplicação (em nosso exemplo: **com.helloworld.helloworld.controller**).

2. Na sequência, clique na opção **New → Class**



3. Na janela **New Java Class**, no item **Name**, digite o nome da Classe (em nosso exemplo: **HelloWorldController**), como mostra a figura abaixo:



4. Clique no botão **Finish** para concluir.

Agora vamos criar o código da Classe Controladora **HelloWorldController**, igual a figura abaixo:

```java
package com.helloworld.helloworld.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/hello-world")
public class HelloWorldController {

    @GetMapping
    public String helloWorld() {
        return "Hello World!!!";
    }

}
```

Na **linha 07** a anotação **@RestController** define que a classe é do tipo controladora rest, que receberá requisições que serão compostas por:

- **URL:** Endereço da requisição (/hello-world)
- **Verbo:** Define qual método HTTP será acionado na Classe controladora.
- **Corpo da requisição (Request Body):** Objeto que contém os dados que serão criados ou atualizados.

Após receber e processar a requisição, a Classe Controladora Responderá a estas requisições com:

- Um **Código de Status HTTP** pertinente a operação que está sendo realizada
- O resultado do processamento (Dados de uma consulta, por exemplo) inserido diretamente no corpo da resposta (**Response Body**)

Na **linha 08** a anotação **@RequestMapping** é usada para mapear solicitações para os métodos da classe controladora **HelloWorldController**, ou seja, definir a **URL** (endereço) padrão do recurso (em nosso exemplo: **/hello-world**). Ao digitar a url do servidor seguida da url do recurso, o Spring envia a requisição para a Classe responsável pelo recurso com este endereço.

Exemplo: http://localhost:8080/hello-world é o endereço do recurso **hello-world** da Classe Controladora **HelloWorldController**.

Na **linha 11** a anotação **@GetMapping** mapeia solicitações HTTP GET para métodos de tratamento específicos, ou seja, indica que o método helloWorld() responderá a todas as requisições do tipo **HTTP GET**, enviadas no endereço **http://localhost:8080/hello-world**, do recurso hello-world.

Nas **linhas 12-14** o método **helloWorld()** retorna uma mensagem de boas vindas, ou seja, quando o endereço for enviado via Postman ou via Browser (Navegador), será exibida a mensagem de boas vindas Hello World!!!

Para concluir, não esqueça de Salvar o código (**File → Save All**)

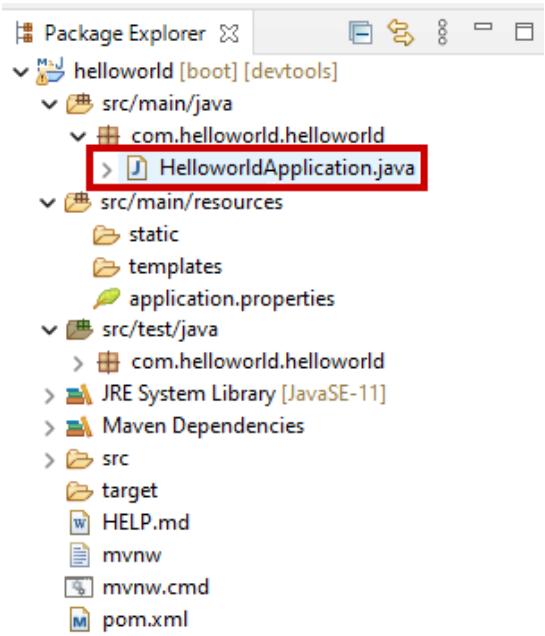
Documentação: HTTP Methods Request
Documentação: HTTP Status Code
Documentação: *@RestController*
Documentação: *@RequestMapping*

[Documentação: @GetMapping]

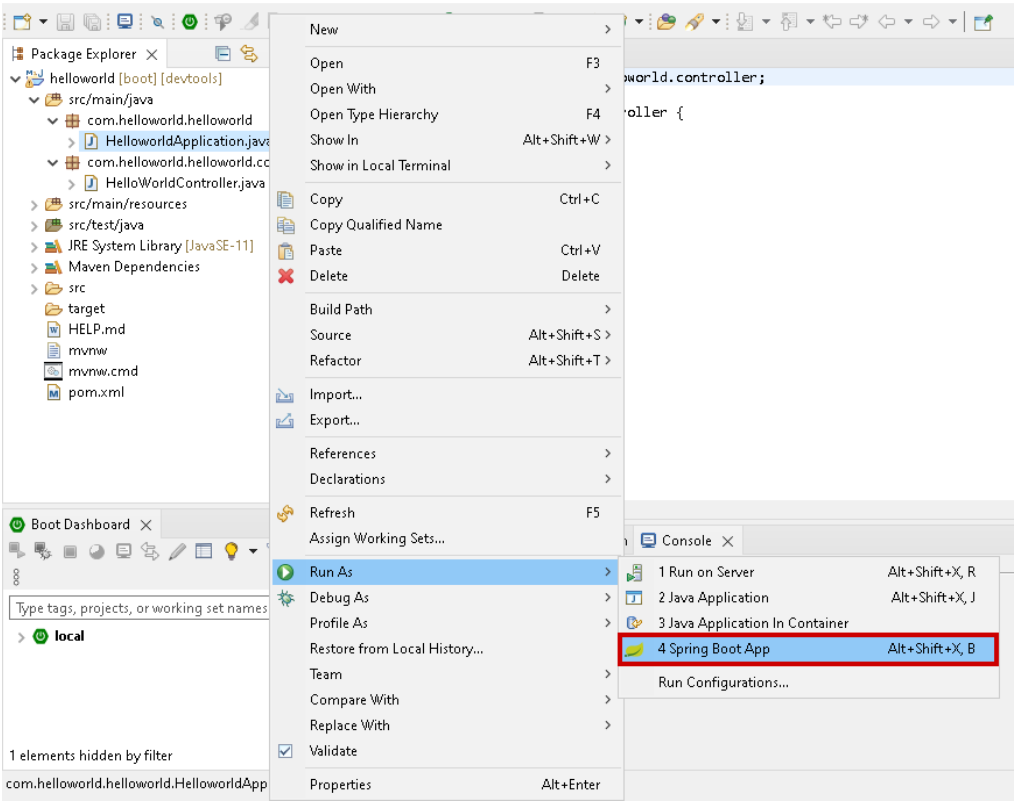
[Código fonte do projeto]

Passo 06 - Executar o Projeto

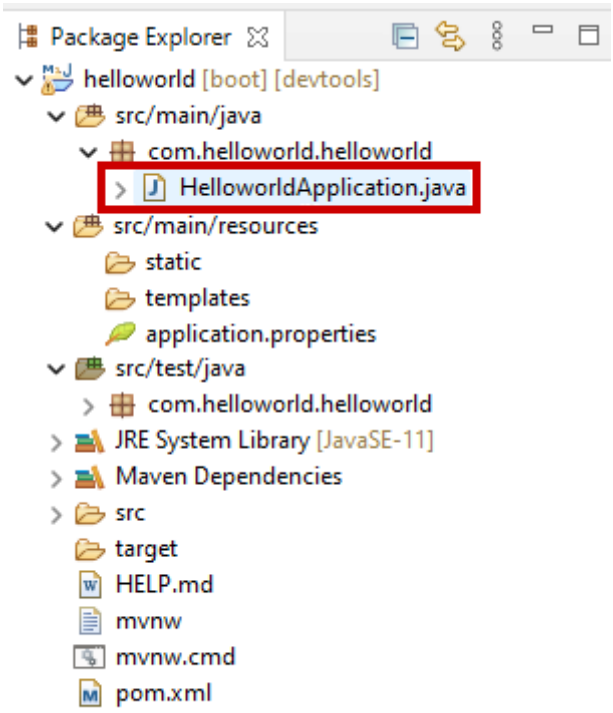
1. No **STS**, na **Package Explorer**, clique na pasta **src/main/java** e na sequência clique no pacote principal **com.helloworld.helloworld**.
2. Clique com o botão direito do mouse sobre o arquivo **HelloworldApplication.java**.



3. No menu que será aberto, clique na opção **Run AS → Spring Boot App** como mostra a figura abaixo:

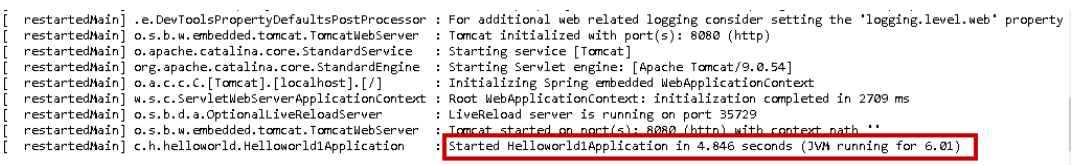


4. Observe que será aberta a janela Console com o log da execução do código.



5. Neste momento o STS varre todo o seu projeto procurando por **TODAS** as Classes que contenham métodos main e checará se o código está correto ou se falta alguma Dependência.

6. Caso esteja tudo certo, o Console exibirá ao final do processamento a mensagem: **Started Helloworld1Application in 4.846 seconds (JVM running for 6.01)**, como mostra a figura abaixo:



Passo 07 - Testar no Insomnia

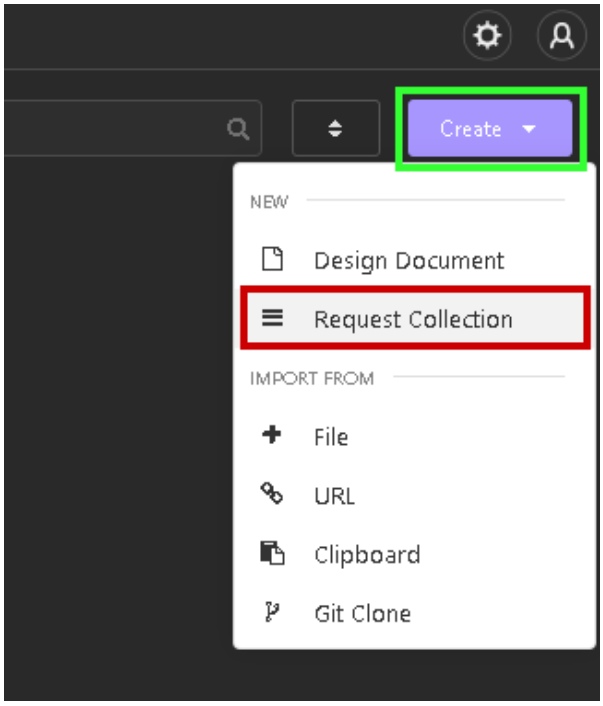
Para testar a aplicação, utilizaremos o **Insomnia**.

Site Oficial do Insomnia

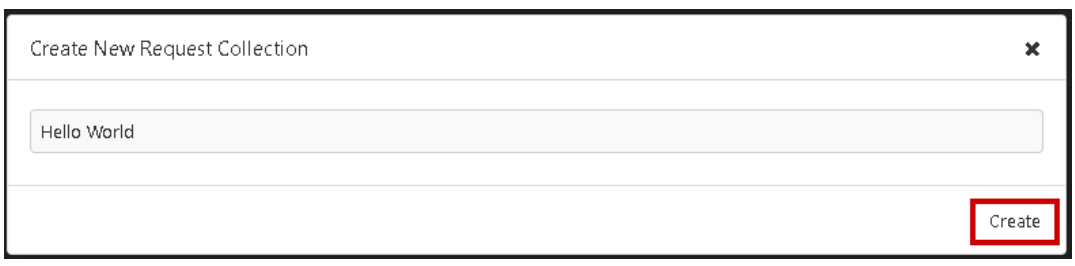
	**DICA:** ***Caso você tenha alguma dúvida quanto a instalação do Insomnia, consulte o Guia de Instalação do Insomnia.***
O Insomnia é um API Client que simplifica o processo de desenvolvimento do Backend de uma aplicação WEB. O Insomnia permite criar, compartilhar e testar aplicações WEB de forma simples e prática. O Insomnia permite aos usuários criar e salvar requisições **HTTP e HTTPS** (Request) desde as mais simples até as mais complexas, bem como ler as suas respostas HTTP (Response).	

Para testar a nossa API, vamos criar uma requisição do tipo **GET**, onde iremos indicar o servidor (localhost), a porta padrão do Spring (8080) e o endereço do recurso (/hello-world).

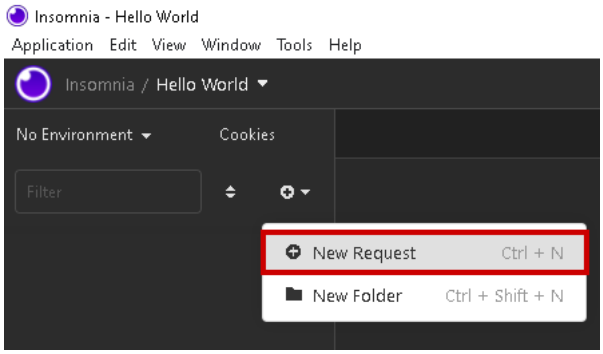
1. Na janela principal do Insomnia, clique no botão **Create** e clique na opção **Request Collection**.



2. Na janela que será aberta, informe o nome da Collection (**Hello World**) e clique no botão **Create** para concluir.

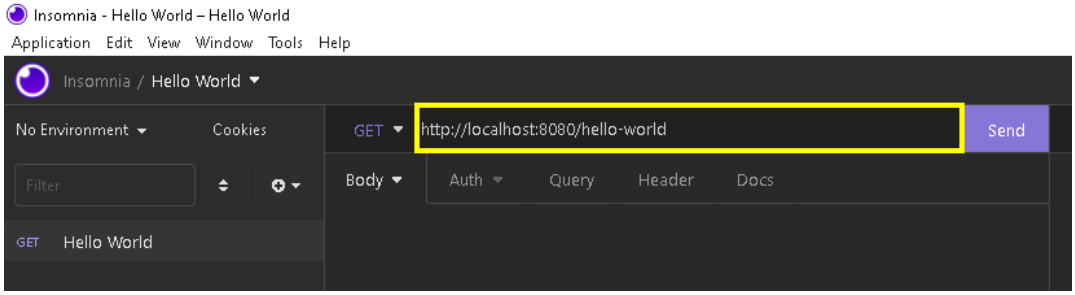


3. Na **Collection Hello World**, clique no botão ⊕. No menu que será aberto, clique na opção **New Request**.



4. Informe o nome da requisição e o Método HTTP que será utilizado (**GET**), indicado na imagem na cor azul. Clique no botão **Create** para concluir.

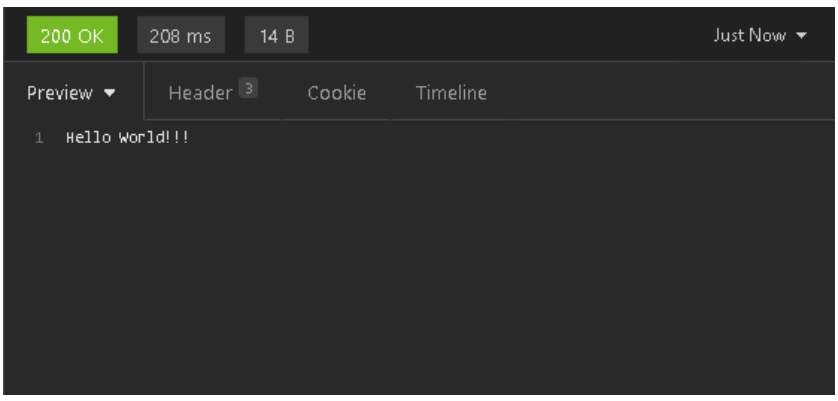
5. Configure a requisição conforme a imagem abaixo:



6. No item marcado em amarelo na imagem acima, informe o endereço da Requisição. A requisição **Hello World** foi configurada da seguinte maneira:

- A primeira parte (http://localhost:8080) é o endereço do servidor local.
- A segunda parte é o **endpoint** configurado na anotação ***@RequestMapping*** (**/hello-world**).

7. Para testar a requisição, com a aplicação rodando, clique no botão Send.
8. O resultado da requisição você confere na imagem abaixo:



Se a mensagem de boas vindas: **Hello World !!!** for exibida, a aplicação está funcionando corretamente.