Primeiros passos com Spring BOOT

Pré-requisitos

- Já sabemos que um projeto WEB é um projeto desacoplado FRONT-BACK
- O Front consome dados do BACK através de chamadas via HTTP (seja em Angular com o HTTPClient ou Javascript com o Fetch).
- Vamos programar em JAVA!

Antes de mais nada...

O que é MAVEN?

- **gerenciador de dependências** - é ele quem faz os downloads das bibliotecas que você vai precisar no seu projeto (claro que, obviamente, você informando em um arquivo de configuração)
- **repositório central** - todas as bibliotecas estão em um servidor na nuvem (mantido pelo próprio MAVEN), chamado Maven Central, facilitando e centralizando o download (você não tem que ficar caçando bibliotecas no google)
- **automatizador de tarefas** - um projeto que tem muitas bibliotecas e muitas dependências tem alguns "problemas" no seu dia-a-dia (manter as bibliotecas atualizadas, fazer todo o *build* da sua aplicação, fazer alguns testes, etc). Então o MAVEN possui scripts prontos para automatizar tudo isso.

Como funciona um projeto SPRING BOOT?

Basicamente:

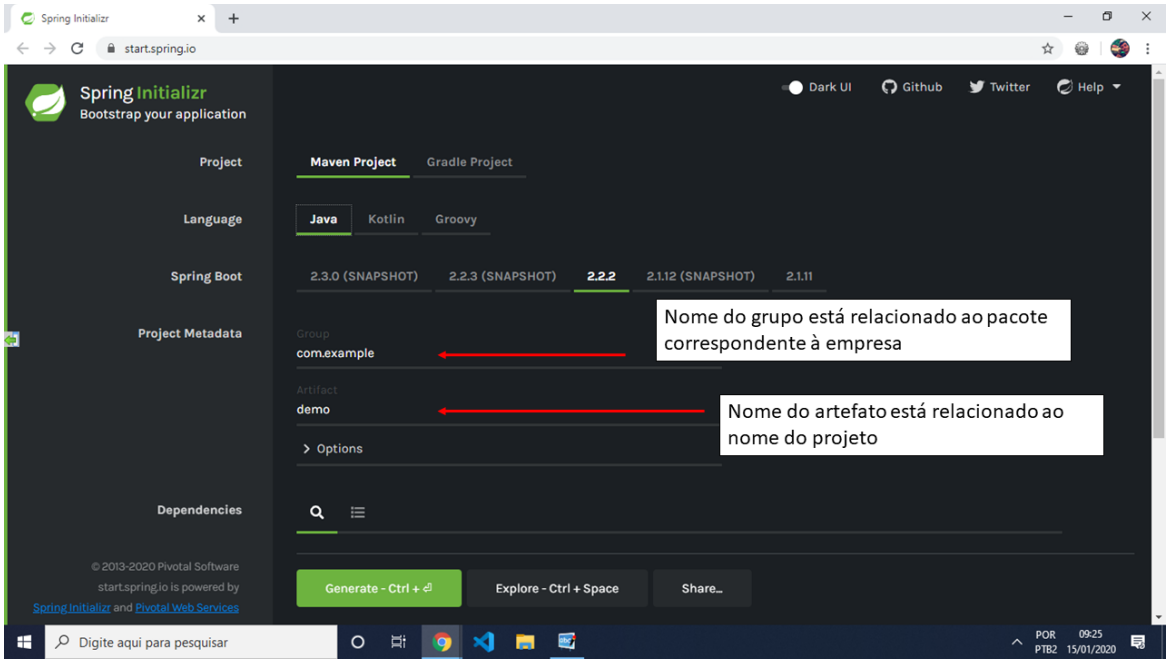
- 1 Classe Principal (com método main mesmo), que vai iniciar um servidor WEB (o TOMCAT) e ele vai gerenciar todas as URLs disponíveis
- Cada URL deve ser mapeada para um determinado método de uma classe. É a execução desse método que dá o retorno da resposta quando acionamos a URL.
- A partir daí, criamos nossos objetos que implementarão nossas lógicas.

Como planejar um projeto SPRING BOOT?

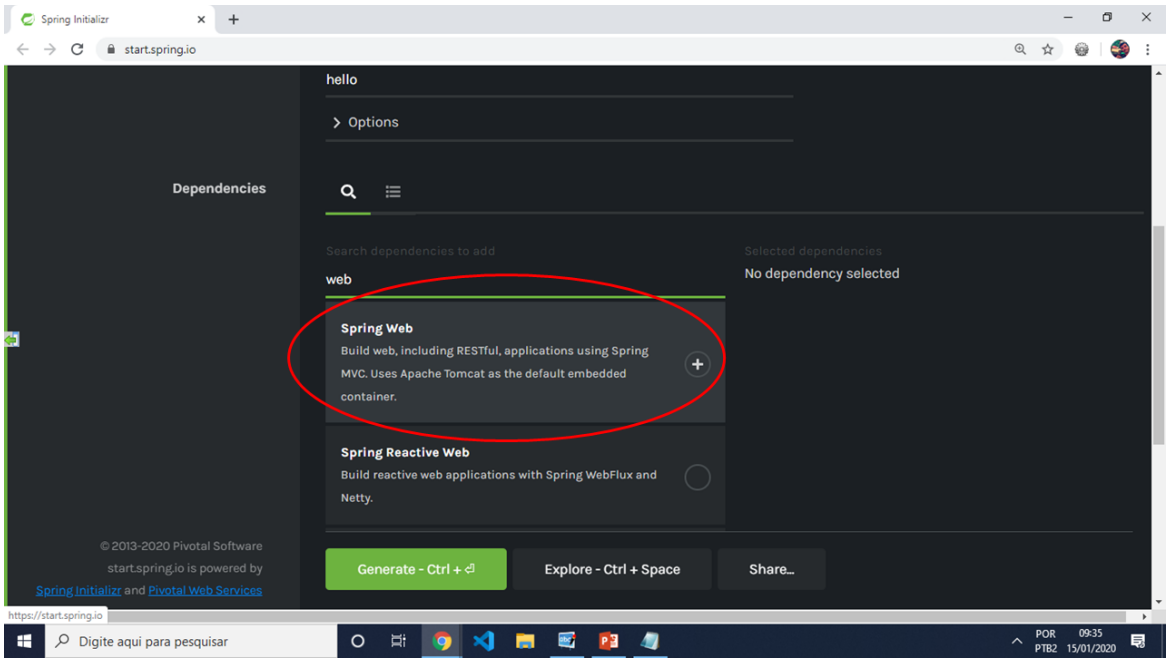
- Quais ENDPOINTS vamos oferecer? (Um Endpoint é uma URL associada a um método do protocolo HTTP - GET, POST, PUT, DELETE).
    - Em geral, temos 1 endpoint para cada objeto do nosso modelo de negócios
    - Objeto de Negócios: PRODUTO
        - URL para recuperar dados de um produto (GET)
        - 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 isso mais adiante.

Mão na massa

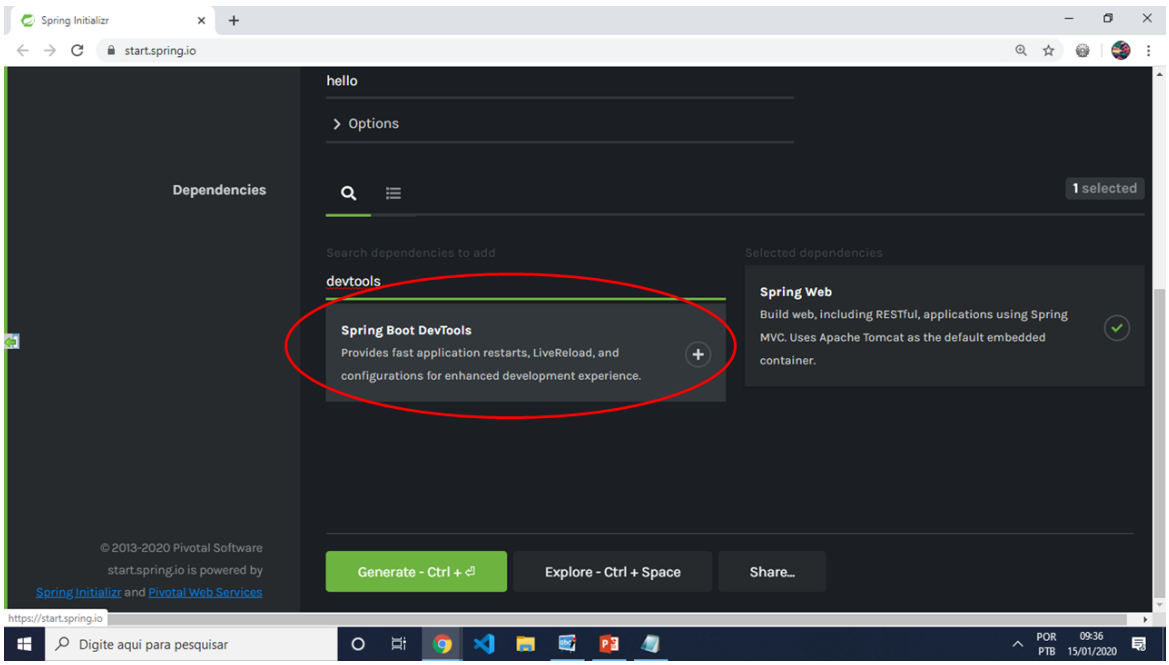
- Acessar o gerador de templates do Spring Boot
    - http://start.spring.io
- Definir o grupo e o artefato do nosso projeto



- Adicionar o plugin base para o SpringBoot (SpringWEB)

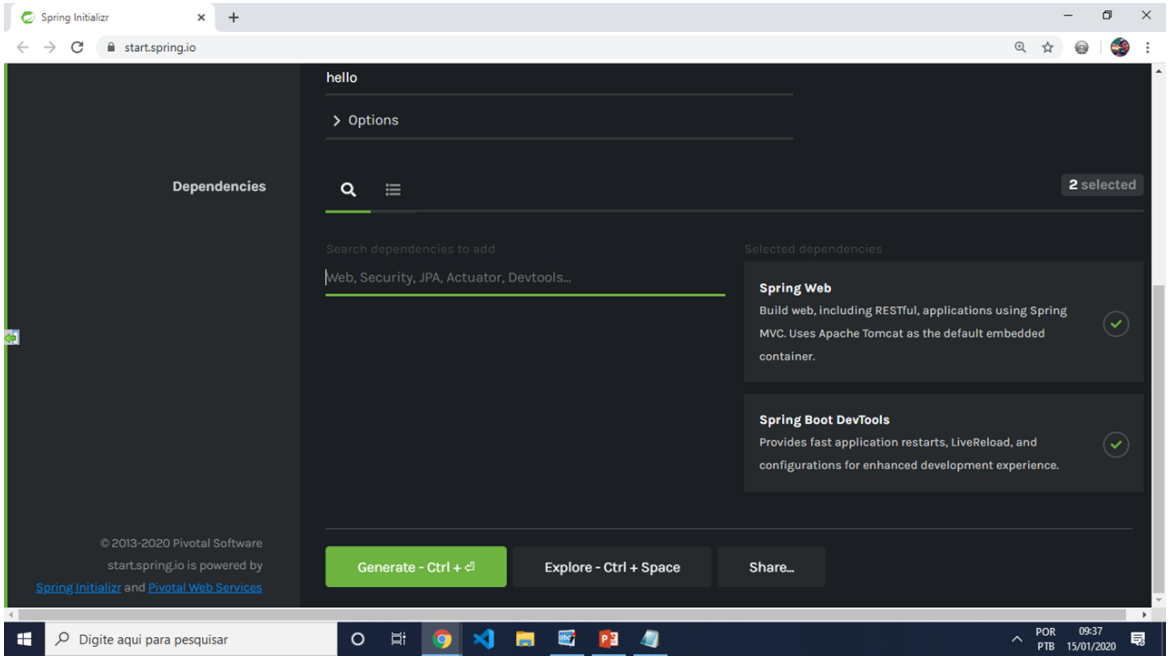


- Adicionar a dependência do DevTools (que poderá nos facilitar com o LiveReload acelerando o desenvolvimento).

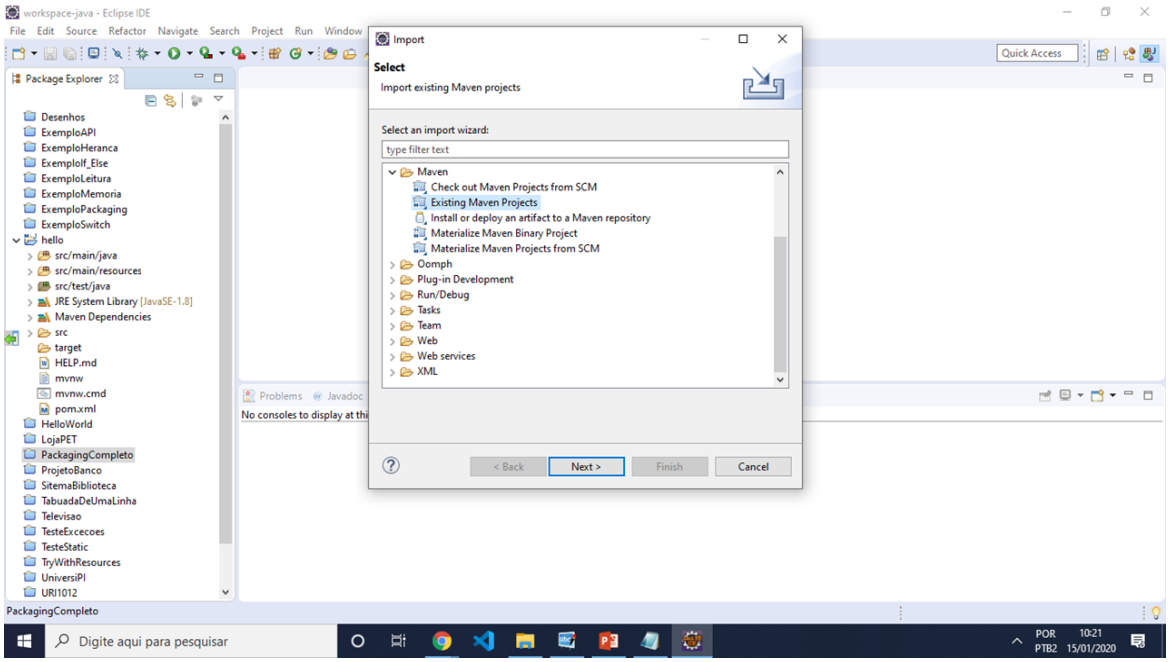


-

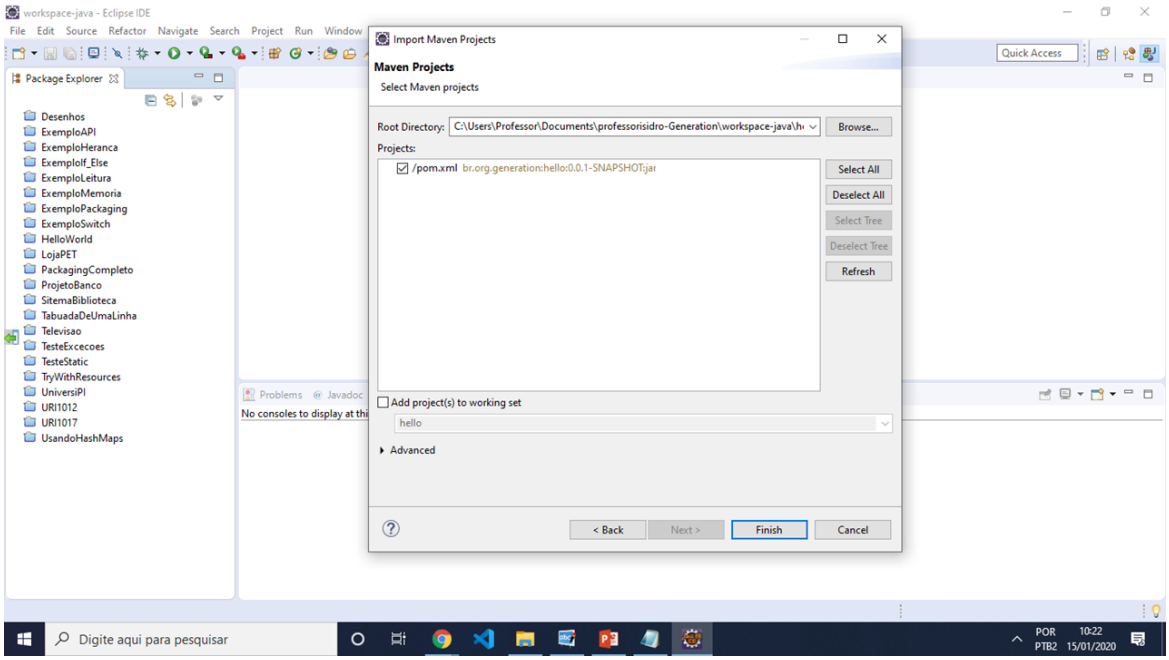
- Ao final, a especificação do projeto deverá estar deste jeito aqui:



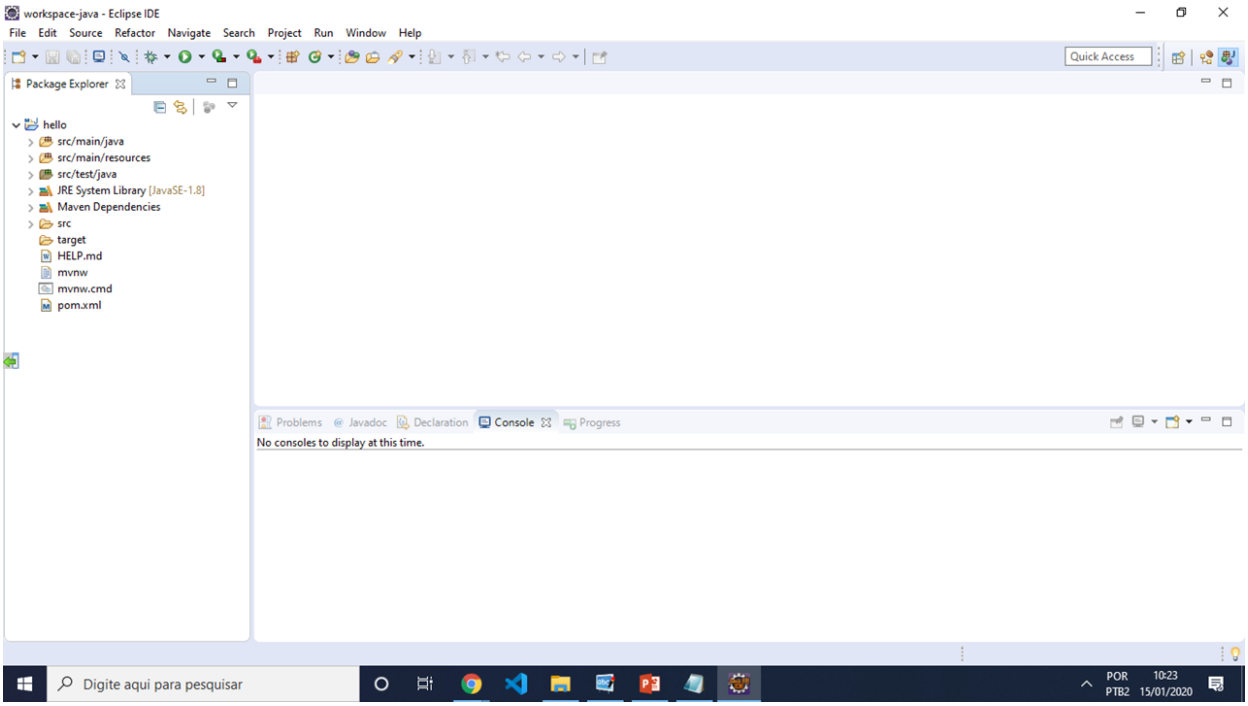
- Importar o projeto no Eclipse (Importar como um Existing Maven Project)



- Selecionar a pasta do projeto e clicar em FINISH



A estrutura do nosso projeto importado para o **eclipse** fica de acordo com a imagem:



Criando nosso primeiro controller

Lembrando:

- Todos os pacotes do nosso projeto deverão estar contidos dentro do pacote-base (formado pelos identificadores do grupo e do artefato)
    - ex: pacote controller
        - `br.org.generation.hello.controller`

Vamos criar nosso primeiro controller

- Criar o pacote que irá conter os controllers
- Criar uma nova classe que chamaremos aqui de `ControllerTeste` com o seguinte conteúdo;

```
package br.org.generation.hello.controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ControllerTeste {

    @GetMapping("/hello")
    public String sayHello() {
       return "Hello World! Nosso primeiro projeto Spring Boot";
    }
}
```

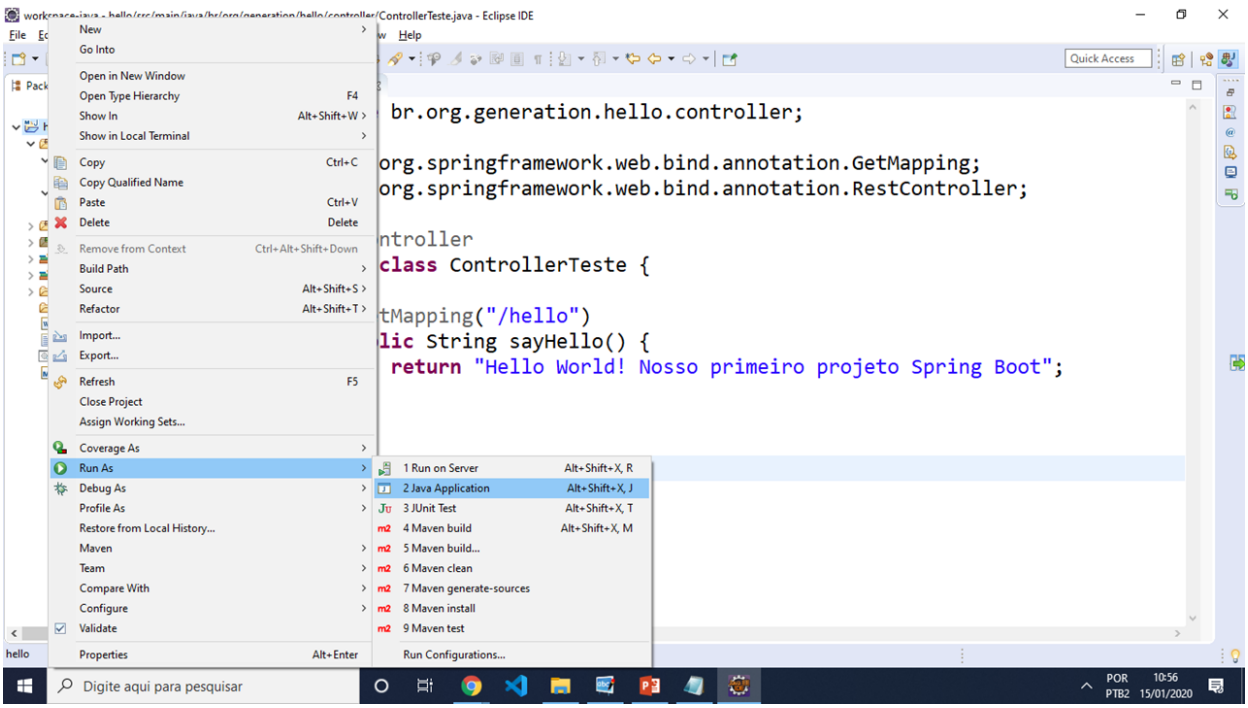
Nesta Classe:

- A anotação `@RestController` define que a classe irá ser acionada a partir de URL (ela começa a atender endpoints)
- O método `sayHello` retorna uma mensagem de boas vindas e é anotado com `@GetMapping("/hello")`. Isso significa que o acesso será via URL pelo método GET através do caminho http://meuservidor/hello

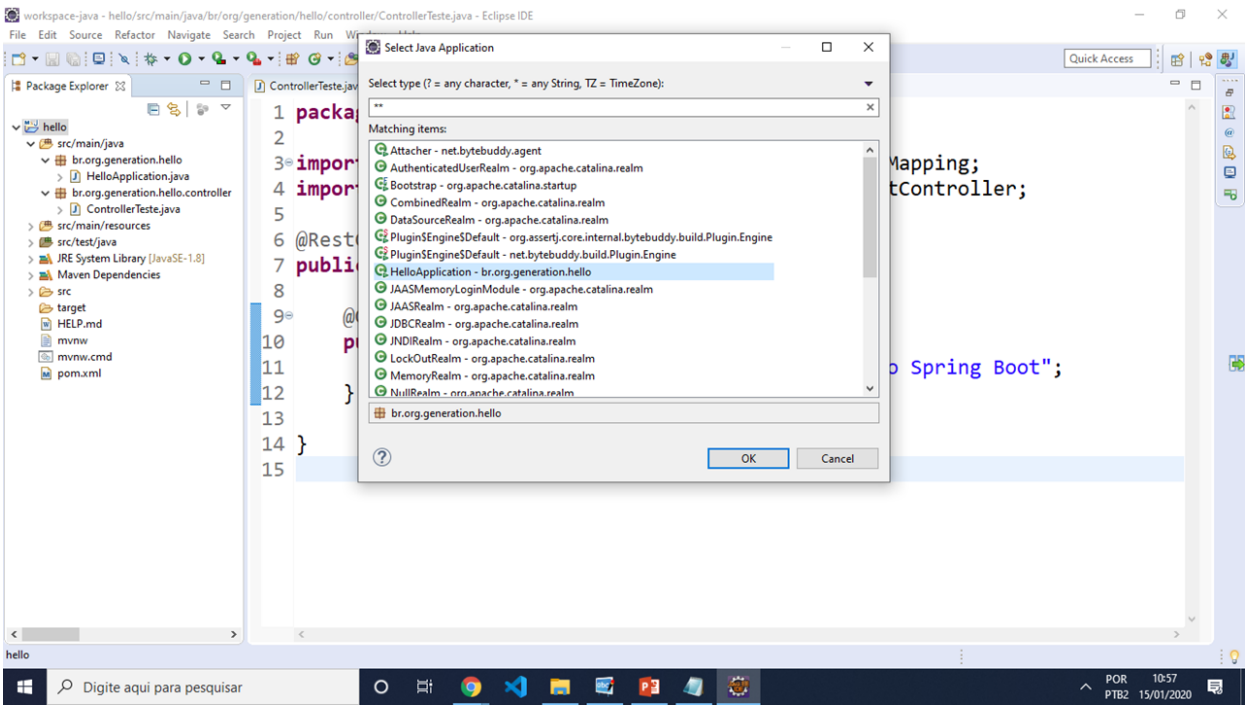
E para executar?

Basta selecionar o projeto, ir com o botão direito do Mouse na seguinte opção

- **Run As > Java Application** como mostra a figura abaixo



Neste ponto será mostrada uma tela uma única vez (sempre na 1a execução) que é para você selecionar a classe que corresponde à sua classe principal. O Eclipse varre todo seu projeto procurando por TODAS as classes que contenham métodos **main**, por isso é importante que você selecione a classe correta. Neste nosso caso, a classe se chama **HelloApplication**.



Vamos testar!

Para testar, basta indicar o servidor (no nosso caso, servidor local, portanto **localhost**) a porta que o SpringBoot trabalha (por padrão porta **8080**, mas é possível mudar) e o

caminho que quermos (neste caso **/hello**).

```
http://localhost:8080/hello
```