

Tecnologias Para Back-End

Prof. JUNIO FIGUEIRÊDO

JUNIOINF@GMAIL.COM

AULA 01 – CRIAÇÃO DO PROJETO

Spring

Spring



Conceitos Introdutórios

Temos que entender o que é um **FRAMEWORK**

- Não será preciso reinventar a roda e gastar tempo com tarefas básicas de desenvolvimento, como criar classes, manipular objetos e definir funções típicas, ou seja **é um conjunto de código pronto**





O que é **SPRING**

- Spring é um **framework** para desenvolvimento de aplicações em **Java**, criado em meados de **2002** por **Rod Johnson**, que se tornou **bastante popular** e adotado ao redor do mundo **devido a sua simplicidade e facilidade de integração com outras tecnologias**.



O que é **SPRING**

- Existem diversos módulos no Spring, cada um com uma finalidade distinta, como por exemplo: o **módulo MVC**, para desenvolvimento de aplicações Web e API's Rest; o **módulo Security**, para lidar com controle de autenticação e autorização da aplicação; e o **módulo Transactions**, para gerenciar o controle transacional.
- <https://spring.io/projects/>



O que é **SPRING BOOT**

- Para resolver dificuldades de criação e configuração de um projeto/módulo do Spring, foi desenvolvido o **Boot**, em meados de 2014, com o propósito de agilizar a criação de um projeto que utilize o Spring como framework, bem como simplificar as configurações de seus módulos.
- O lançamento do Spring Boot foi um marco para o desenvolvimento de aplicações Java, pois tornou tal tarefa mais simples e ágil de se realizar, facilitando bastante a vida das pessoas que utilizam a linguagem Java para desenvolver suas aplicações.



O que é **SPRING BOOT**

- A versão 3 do Spring Boot foi lançada em novembro de 2022, trazendo algumas novidades em relação à versão anterior. Dentre as principais novidades, se destacam:
 - . Suporte ao Java 17
 - . Migração das especificações do Java EE para o Jakarta EE
 - . Etc..

Spring



Tecnologias e Objetivos



Objetivos

- O objetivo neste curso é usarmos o **Spring Boot** para **desenvolvermos uma API Rest**, com algumas funcionalidades.
- A ideia é desenvolver um **CRUD**, sendo as quatro operações fundamentais das aplicações: **cadastro, listagem, atualização e exclusão de informações**.
- Vamos ver também como **aplicar validações das informações** que chegam na nossa API, usando o *Bean Validation*.
- Depois, vamos aprender a **utilizar o conceito de paginação e ordenação** das informações que a nossa API vai devolver.

Tecnologias

- Spring Boot 3
- Java 21
- Lombok
- MySQL/ Flyway
- JPA/Hibernate
- Maven
- Insomnia

SPRING BOOT 3 e JAVA 21

- Faremos tudo isso usando algumas tecnologias, como Spring Boot 3, sendo a última versão disponibilizada pelo framework.
- Usaremos, também, o **Java 21** sendo a última versão **LTS** (**Long-term support**, em português "Suporte de longo prazo") que possui maior tempo de suporte disponível para o Java

LOMBOK

- Utilizaremos em conjunto com o projeto o **Lombok**, responsável por fazer a geração de códigos repetitivos, como *getters*, *setters*, *toString*, entre outros. Tudo via anotações para o código ficar menos verboso.

Conceitos Introdutórios

MySQL e Flyway

- Usaremos o banco de dados **MySQL** para armazenar as informações da API e junto com ele utilizaremos a biblioteca **Flyway**. Isso para termos o controle do histórico de evolução do banco de dados, um conceito que chamamos de **Migration**.

JPA e Hibernate

- A camada de persistência da nossa aplicação será feita com a **JPA** (*Java Persistence API*), com o **Hibernate** como implementação dessa especificação e usando os módulos do Spring Boot, para tornar esse processo o mais simples possível.

Maven

- Usaremos o **Maven** para gerenciar as dependências do projeto, e também para gerar o *build* da nossa aplicação. Por último, como focaremos na **API Rest** (apenas no Back-end), **não** teremos **interface gráfica**, como páginas **HTML** e nem **Front-end** e **aplicativo mobile**.

Insomnia

- Mas **para testarmos a API**, usaremos o **Insomnia**, sendo uma ferramenta usada para testes em API. Com ela, conseguimos simular a requisição para a API e verificar se as funcionalidades implementadas estão funcionando.

Spring

Clinica Médica



Médicos(as)



Pacientes



Consultas

Conceitos Introdutórios

Clinica Médica

- Trabalharemos em um projeto de uma clínica médica fictícia. Temos uma empresa chamada Voll Med, que possui uma clínica que precisa de um aplicativo para monitorar o cadastro de médicos, pacientes e agendamento de consultas.
- Esse é o nosso objetivo neste curso, aprender a usar o Spring Boot na versão 3 para desenvolvermos o projeto dessa clínica médica, utilizando as tecnologias mencionadas anteriormente.

Spring



INICIANDO O PROJETO



Projeto:

- Spring Initializr
- Estrutura do projeto
- Primeira Página de Teste

Projeto:

- Acessaremos o Spring Initializr pelo site <https://start.spring.io/>



Spring

Conceitos Introdutórios

Spring Initializr:

Project

☐ Gradle - Groovy

☐ Gradle - Kotlin

☒ Maven

Language

☒ Java

☐ Kotlin

☐ Groovy

Spring Boot

☐ 3.3.0 (SNAPSHOT)

☒ 3.3.0 (M1)

☐ 3.2.3 (SNAPSHOT)

☐ 3.2.2

☐ 3.1.9 (SNAPSHOT)

☐ 3.1.8

Project Metadata

Group med.voll

Artifact api

Name api

Description API Rest da aplicação Voll.med

Package name med.voll.api

Packaging ☒ Jar ☐ War

Java ☒ 21 ☐ 17

- Depois de preencher os campos, vamos inserir as dependências.

- Clicar no botão "Add dependencies".

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Inicializar:

- Será aberta uma pop-up com diversas dependências do Spring Boot, e do Spring para selecionarmos.
- Vamos apertar a tecla "Ctrl" do teclado e clicar em cada uma das dependências que desejamos adicionar, sendo elas:
 - Spring Boot DevTools
 - Lombok
 - Spring Web

Spring

Conceitos Introdutórios

Spring Initializr:

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Boot DevTools

DEVELOPER TOOLS

Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Lombok

DEVELOPER TOOLS

Java annotation library which helps to reduce boilerplate code.

Spring Web

WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring

Conceitos Introdutórios

Spring Initializr:

- Informações e adicionarmos as dependências, podemos selecionar o botão "Generate" na parte inferior da página.



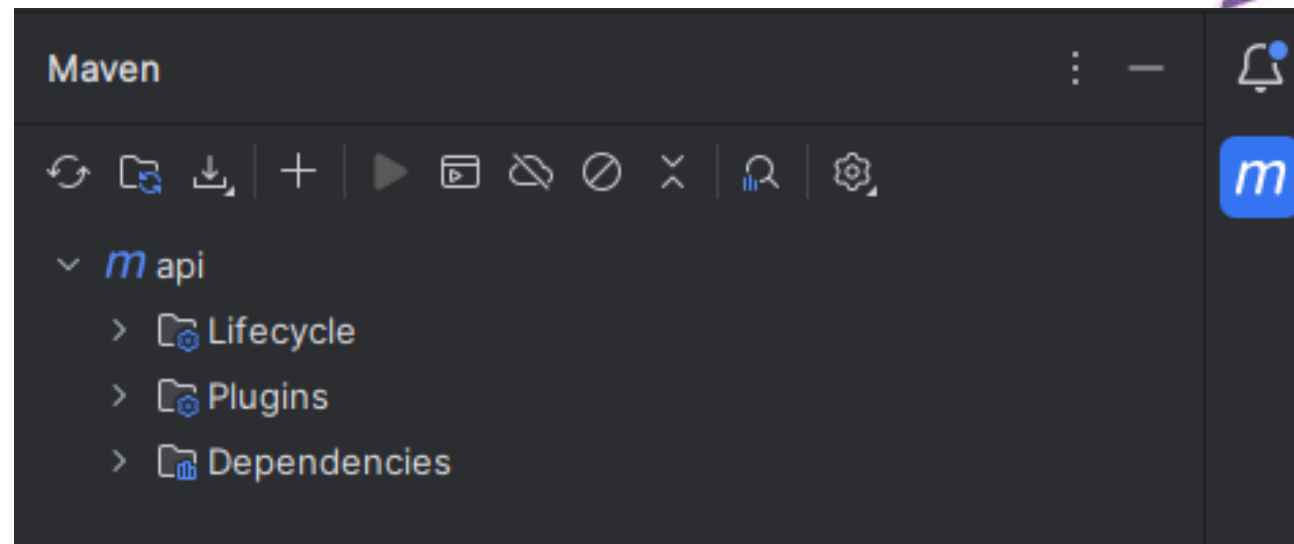
GENERATE CTRL + 

- Dessa forma, **vamos gerar** o projeto e teremos **um arquivo .zip** com o **projeto compactado**.
- Descompactar o arquivo api.zip para dentro da pasta api.

Conceitos Introdutórios

Estrutura do Projeto

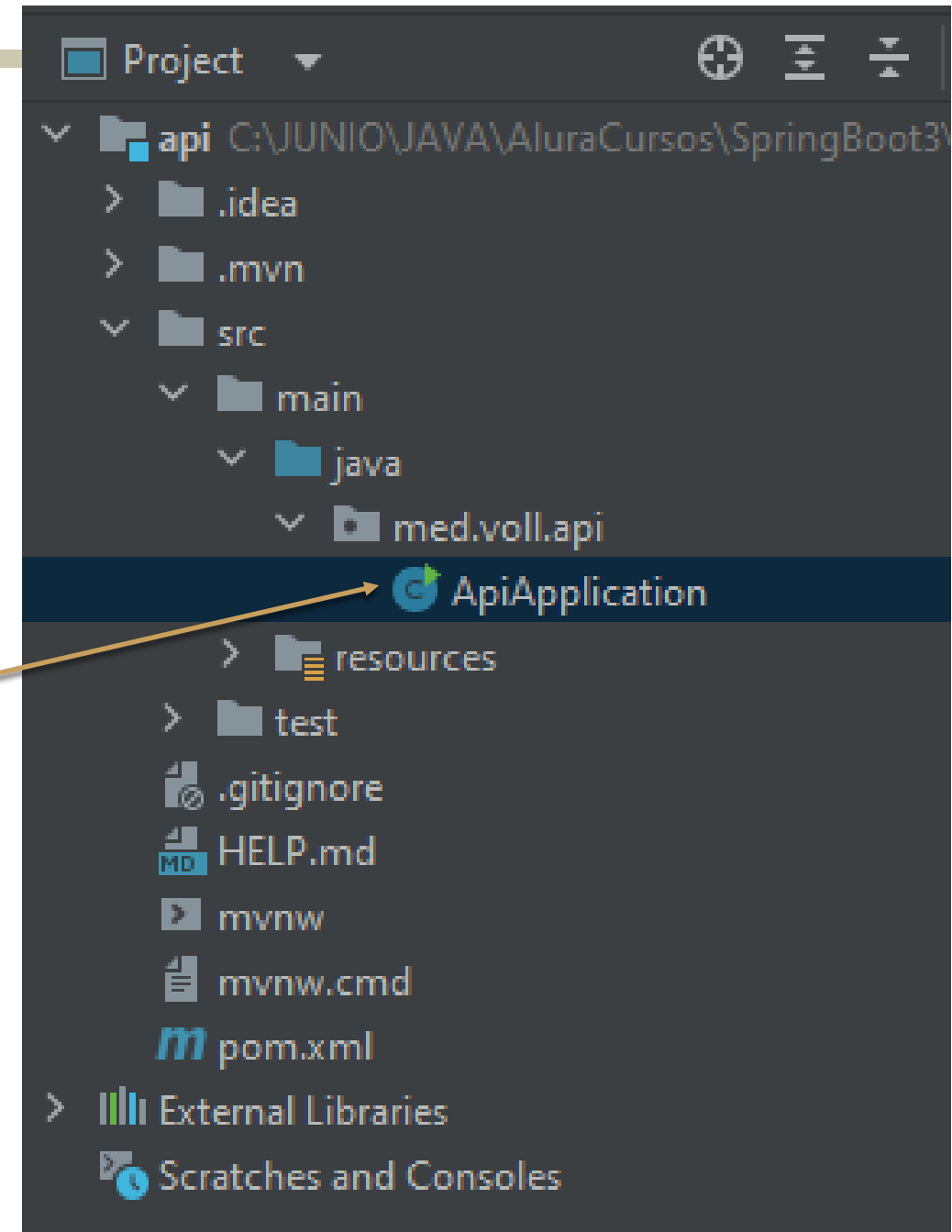
- Vamos utilizar o IntelliJ.
- Vamos importar o arquivo api descompactado
- Uma vez importado, visualize as dependências



Spring

Estrutura do Projeto

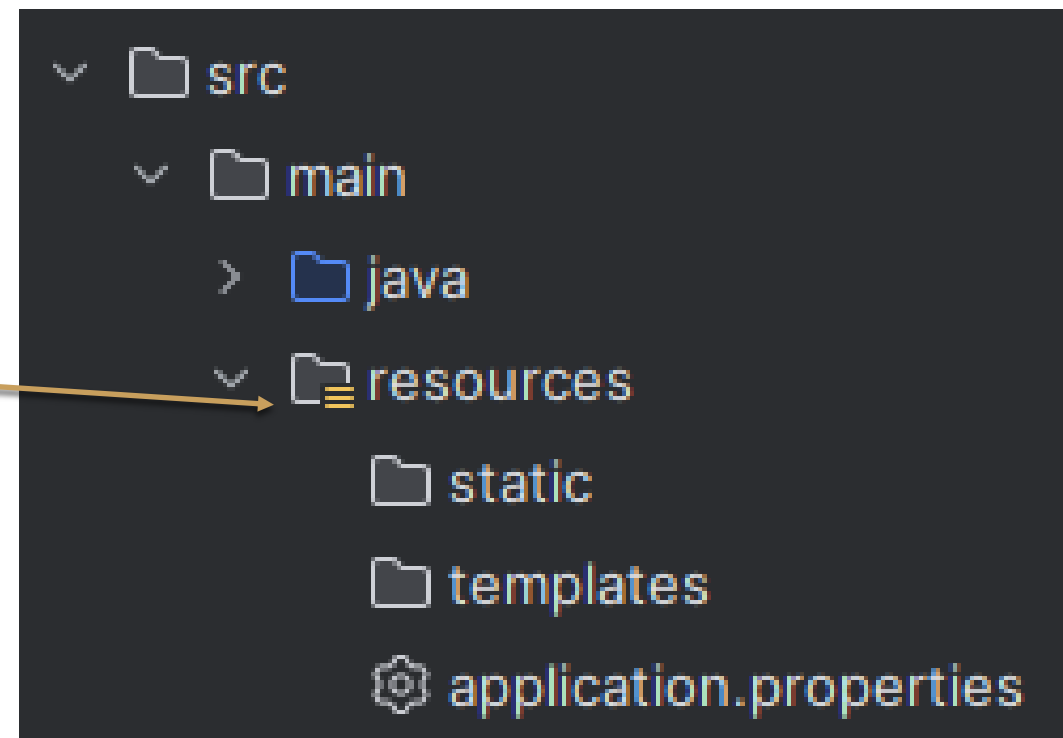
A classe principal do spring boot



Estrutura do Projeto

Diretório do Maven que ficam os arquivos e as configurações do projeto

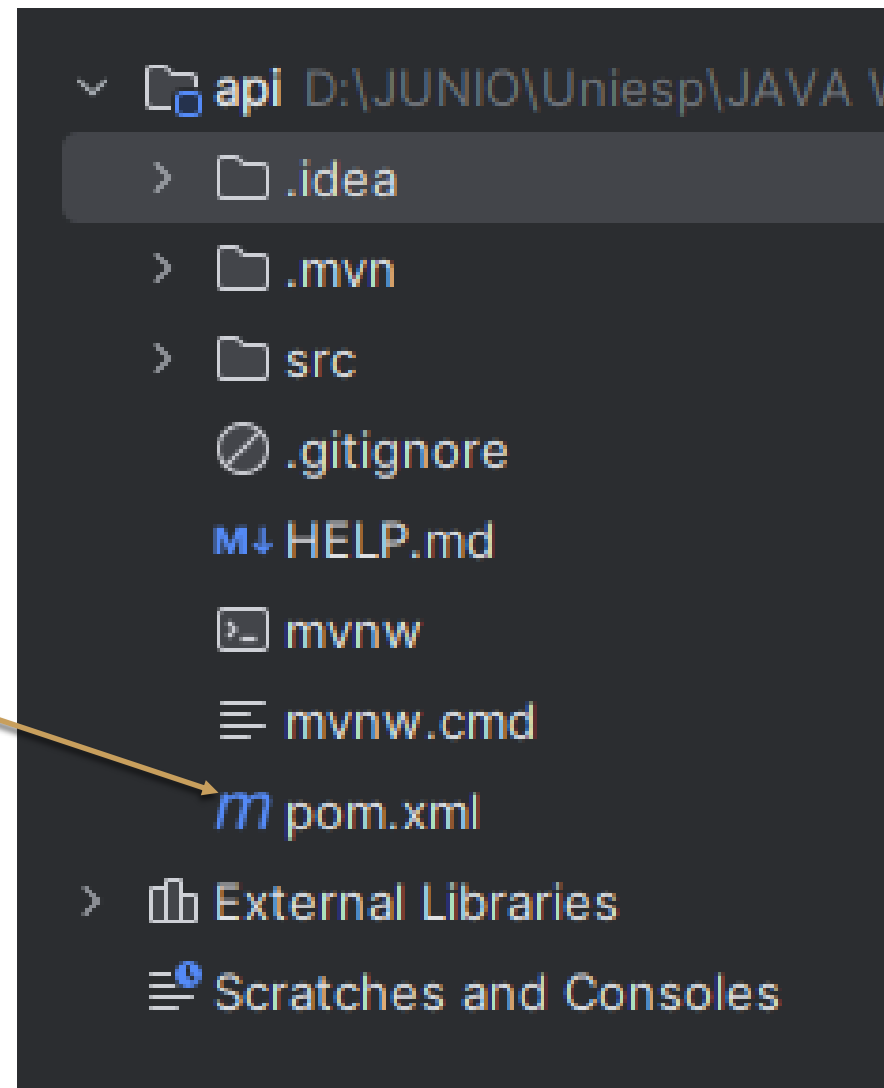
- **Static** temos os arquivos JS, imagens e etc..
- **Templates** estão os templates HTML, as páginas do projeto...
- **application.properties** Esse é a pasta de configurações do projeto com Spring Boot, usaremos bastante esse arquivo.



Estrutura do Projeto

pom.xml é configurado corretamente, com as dependências que escolhemos.

Gera as configurações do projeto, como vimos em **resources** e já cria a classe **main ApiApplication**, com a estrutura inicial para rodarmos a nossa aplicação



Spring



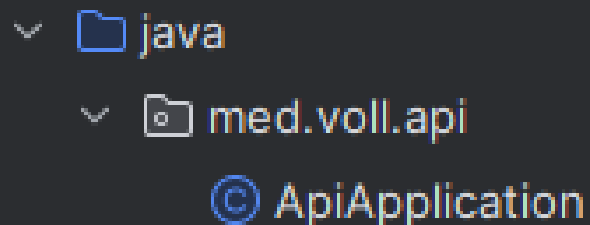
RODANDO A APLICAÇÃO



Conceitos Introdutórios

Rodando a Aplicação

- Para executarmos esse projeto com Spring Boot, precisamos rodar a classe **ApiApplication** que possui o **método main**.



```
▼ java
  ▼ med.voll.api
    © ApiApplication
```

```
@SpringBootApplication
public class ApiApplication {

    public static void main(String[] args) {
        SpringApplication.run(ApiApplication.class,
    }
```

Conceitos Introdutórios

Rodando a Aplicação

- Como funcionava para rodarmos uma aplicação web?
- Nós adicionávamos um servidor de aplicações, como **TomCat**, **Jetty**, **Glassfish** e **Weblogic**, e colocávamos o projeto dentro do servidor e o inicializávamos.
- Dessa forma, ele fazia toda configuração e disponibilizava a aplicação.



Rodando a Aplicação

- No Spring Boot, o processo foi invertido.
- Ao invés de termos um servidor e colocarmos dentro dele a aplicação.
- Temos agora dentro da aplicação, um de aplicação.
- Por padrão, o projeto vem com o TomCat como servidor de aplicação e ele já está embutido dentro das dependências do módulo web.
- Ele não aparece no arquivo pom.xml porque está no xml do Spring Boot herdado.

Spring

Conceitos Introdutórios

Rodando a Aplicação

```
public class HelloController {  
}
```

- Vamos criar um **cotroller/classe** chamada de **HelloController**.
- Para que as classes não fiquem soltas dentro do **pacote api**, vamos criar um SubPacote chamado **controller**

```
package med.voll.api.controller;
```
- No **controller**, é necessário chamarmos algum método.
- Criaremos o método **public string olaTurma(){}.**
- Dentro dele, vamos retorna a **string** “**Olá Turma!**”.

Rodando a Aplicação

```
package med.voll.api.controller;  
  
no usages  
  
public class HelloController {  
    no usages  
  
    public String olaTurma(){  
        return "Olá Turma!!";  
    }  
}
```

Rodando a Aplicação

- Vamos inserir algumas anotações:
 - **@RestController** – define uma classe que contém métodos para uma API RESTful. Isso não é uma **aplicação tradicional**, e sim uma **API REST**.
 - **@RequestMapping** – mapeia requisições REST.

Rodando a Aplicação

-

```
@RestController
@RequestMapping("/hello")
public class HelloController {
    no usages

    public String olaTurma(){
        return "Olá Turma!!";
    }
}
```

Rodando a Aplicação

- Entendendo o `@RequestMapping("/hello")`:
 - Isso para informarmos qual a URL que esse controller vai responder, que será `/hello`. Assim, ao chegar uma requisição para `localhost:8080/hello` vai cair neste controller.
- No controller, é necessário chamarmos algum método.
- Criaremos o método `public String olaTurma(){}` . Dentro dele, vamos retorna a `String "Olá Turma!"`.

Rodando a Aplicação

```
@RestController
@RequestMapping("/hello")
public class HelloController {
    no usages
    @GetMapping
    public String olaTurma(){
        return "Olá Turma!!";
    }
}
```

Spring

Conceitos Introdutórios

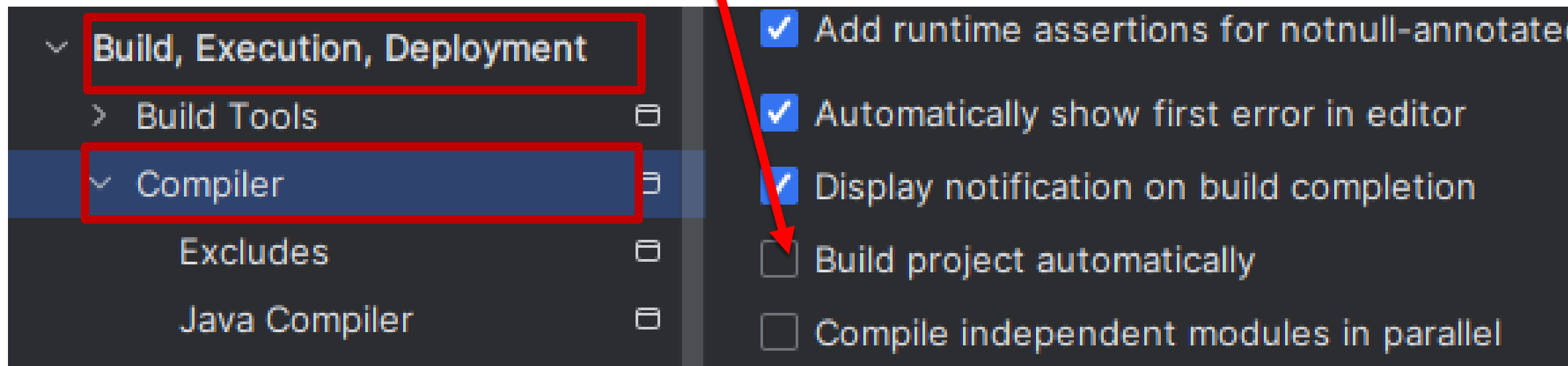
Rodando a Aplicação

- Estamos informando para o Spring que essa classe é um controller, com o `RequestMapping("/hello")`.
- Logo, se chegou uma requisição `/hello` e ela é do tipo `get`, será chamado o método `olaMundo()`.
- Chamar o navegador <http://localhost/hello> teste!!! Veja que apareceu `Olá Turma!!!`
- Altere o return do método para

```
return "Olá Turmasss!!";
```
- Salve e atualize o navegador!!!! O que foi que aconteceu???

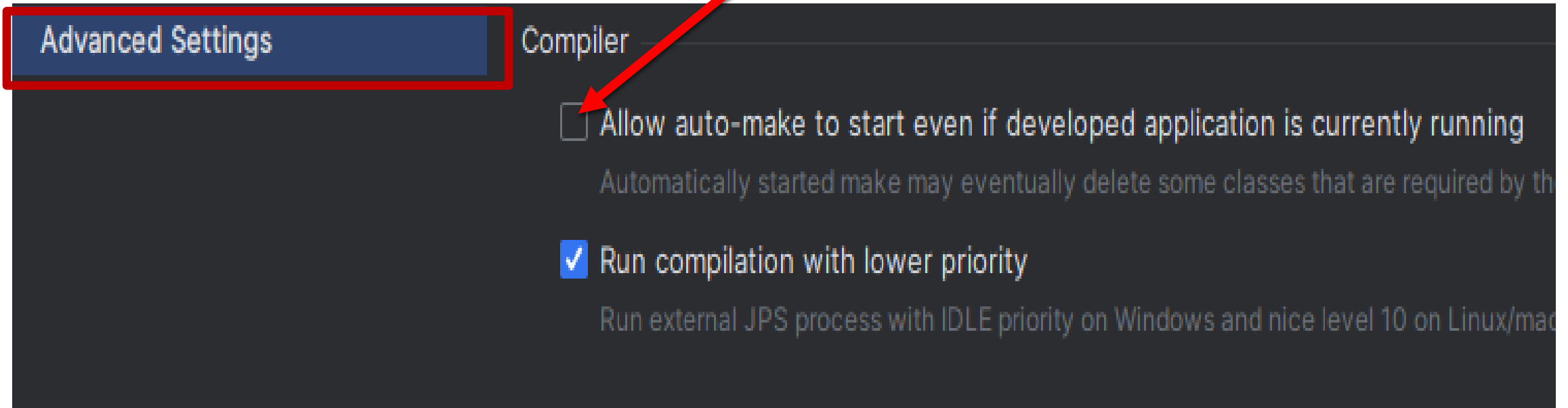
Rodando a Aplicação

- É preciso configurar o projeto para o DevTools funcionar.
- Faça Ctrl + Alt + S, marque essa opção, depois no **botão Apply**



Rodando a Aplicação

- No mesmo menu escolha, marque essa opção, depois no botão Apply, OK



- Para a aplicação e roda novamente. Vamos testar!!!!

Rodando a Aplicação

- Exibimos o Olá Turma e o projeto está configurando.
- Agora, podemos começar a implementar as funcionalidades na aplicação, em que teremos o CRUD dos médicos e pacientes, com agendamentos e cancelamentos das consultas.
- Vamos realizar isso na próxima aula

Spring

DÚVIDAS ????

