@SpringBootApplication

@Controller -> Diz que essa é uma classe responsável por receber requisições web.

@RestController

@GetMapping(“/hello”) -> Caminho

@ResponseBody -> Indica que o retorno deve vir como resposta da requisição.

@Component -> Bean é uma classe onde o programador determina que o Spring container que vai gerenciar as instâncias dessa classe, ficando responsável por criar, gerenciar e destruir todas as instâncias dessa classe.

**Outra forma de definir beans**

@Configuration -> componente Spring com o objetivo de servir para a definição de beans

@Bean -> indica que o método instancia, configura e inicializa um novo objeto que vai ser gerenciado pelo container Spring

(Bean é nome dado para objetos gerenciados pelo container do spring)



Como dizer que uma classe é um BEAN (um componente Spring) e que seus objetos podem ser injetados em outros beans -> @Componente em cima da classe

@AutoWired fornece controle sobre onde e como a ligação entre os beans deve ser realizada. Pode ser usado para em métodos setter, no construtor, em uma propriedade ou métodos com nomes arbitrários e / ou vários argumentos.

@AutoWired(required=false)

[Introdução prática ao Spring Framework com uso de Anotações (devmedia.com.br)](https://www.devmedia.com.br/introducao-pratica-ao-spring-framework-com-uso-de-anotacoes/27859#:~:text=A%20anota%C3%A7%C3%A3o%20%40%20Autowired%20fornece%20controle,arbitr%C3%A1rios%20e%20%2F%20ou%20v%C3%A1rios%20argumentos.)

@Primary

@Qualifier(“email”)

@Rentention(RetentionPolicy.RUNTIME)

@Profile(“prod”)

@PostConstruct

@PreDestroy

@Bean(initMethod = "init", destroyMethod = "destroy")

@Value -> você faz a injeção de um valor usando uma expression do Spring

Ex: @Value("${notificador.email.porta-servidor}")

@ConfigurationProperties

@Entity (name=”txt”) // a propriedade não é obrigatória

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name=”txt”) // a propriedade não é obrigatória