Springboot异常分类

| **特性** | **可受检异常（Checked）** | **不可受检异常（Unchecked）** |
| --- | --- | --- |
| 父类 | Exception | RuntimeException |
| 编译是否强制处理 | ✅ 是 | ❌ 否 |
| 抛出后是否必须 catch 或 throws | ✅ 是 | ❌ 否 |
| 常见场景 | 文件读写、数据库、网络 | 数组越界、空指针、算数异常等 |

@Value 注解在 Spring 中非常常用，它的作用是：从配置文件或环境变量中注入值到 Java 变量中。

Spring Boot 是一个基于 Spring 框架 的快速开发框架，用来简化 Java 后端应用的开发。

## 核心特点总结

| **特性** | **说明** |
| --- | --- |
| ✅ 自动配置 | 帮你自动配置各种组件（数据库、Web服务等） |
| ✅ 内嵌服务器 | 内部自带 Tomcat/Jetty，不用你部署 |
| ✅ 快速启动类 | 一键启动项目，带 main() 方法 |
| ✅ 没有 XML | 几乎不用写 XML 配置 |
| ✅ 约定优于配置 | 只需要少量配置，默认帮你配好大部分东西 |

总结：Spring Boot 核心特性

* 自动配置 根据依赖自动创建和装配 Bean
* 起步依赖 starter打包好了一整套依赖组合

spring-boot-starter-web 用于构建 Web 项目（含 MVC + Tomcat）

spring-boot-starter-data-jpa 数据库操作（基于 JPA）

spring-boot-starter-security 安全框架（权限认证）

spring-boot-starter-thymeleaf 模板引擎

* 内嵌容器 自带 Tomcat，不需部署外部服务器
* 一键运行 Spring Boot 项目最终会打成一个 可运行的 JAR 包，部署简单
* 零 XML 注解配置，干净利落
* 简化配置 统一使用 application.properties文件来进行集中式配置，非常清晰好管理
* 开发者友好 热部署、监控、工具丰富

Springboot定时任务

1. 开启定时任务支持

只需要在你的启动类或者配置类上加上这个注解：

@EnableScheduling

@SpringBootApplication

public class MyApp {

public static void main(String[] args) {

SpringApplication.run(MyApp.class, args);

}

}

2. 编写定时任务方法

在任意一个 Spring 管理的 Bean 中（比如 @Service、@Component 的类），使用 @Scheduled 注解标记你想定时执行的方法：

@Component

public class MyScheduler {

// 每5秒执行一次

@Scheduled(fixedRate = 5000)

public void printHello() {

System.out.println("Hello! 当前时间：" + LocalDateTime.now());

}

}

## 常用的定时配置方式（3种）

| **用法** | **示例** | **说明** |
| --- | --- | --- |
| fixedRate | @Scheduled(fixedRate = 5000) | **每隔5秒执行一次**（上次执行开始算起） |
| fixedDelay | @Scheduled(fixedDelay = 5000) | **上次执行完后隔5秒再执行** |
| cron | @Scheduled(cron = "0 0/5 \* \* \* ?") | 使用 **cron 表达式**，**每5分钟执行一次** |

## Cron 表达式详解（常用）

格式： 秒 分 时 日 月 周 [年（可选）]

| **表达式** | **含义** |
| --- | --- |
| 0/10 \* \* \* \* ? | 每10秒执行一次 |
| 0 0/5 \* \* \* ? | 每5分钟执行一次 |
| 0 0 1 \* \* ? | 每天凌晨1点执行一次 |
| 0 0 9 ? \* MON-FRI | 每周一到周五早上9点执行 |