
黑马程序员™
www.itheima.com
传智播客旗下
高端IT教育品牌
Spring Boot

目录 Contents

SpringBoot 概念

SpringBoot提供了一种快速使用Spring的方式，基于约定优于配置的思想，可以让开发人员不必在配置与逻辑业务之间进行思维的切换，全身心的投入到逻辑业务的代码编写中，从而大大提高了开发的效率，一定程度上缩短了项目周期。2014 年 4 月，Spring Boot 1.0.0 发布。Spring的顶级项目之一(https://spring.io)。

Spring 缺点

1） 配置繁琐

虽然Spring的组件代码是轻量级的，但它的配置却是重量级的。一开始，Spring用XML配置，而且是很多XML配置。Spring 2.5引入了基于注解的组件扫描，这消除了大量针对应用程序自身组件的显式XML配置。Spring 3.0引入了基于Java的配置，这是一种类型安全的可重构配置方式，可以代替XML。

所有这些配置都代表了开发时的损耗。因为在思考Spring特性配置和解决业务问题之间需要进行思维切换，所以编写配置挤占了编写应用程序逻辑的时间。和所有框架一样，Spring实用，但它要求的回报也不少。

2）依赖繁琐

项目的依赖管理也是一件耗时耗力的事情。在环境搭建时，需要分析要导入哪些库的坐标，而且还需要分析导入与之有依赖关系的其他库的坐标，一旦选错了依赖的版本，随之而来的不兼容问题就会严重阻碍项目的开发进度。

SpringBoot 功能

1） 自动配置

Spring Boot的自动配置是一个运行时（更准确地说，是应用程序启动时）的过程，考虑了众多因素，才决定Spring配置应该用哪个，不该用哪个。该过程是SpringBoot自动完成的。

2） 起步依赖

起步依赖本质上是一个Maven项目对象模型（Project Object Model，POM），定义了对其他库的传递依赖，这些东西加在一起即支持某项功能。

简单的说，起步依赖就是将具备某种功能的坐标打包到一起，并提供一些默认的功能。

3） 辅助功能

提供了一些大型项目中常见的非功能性特性，如嵌入式服务器、安全、指标，健康检测、外部配置等。

Spring Boot 并不是对 Spring 功能上的增强，而是提供了一种快速使用 Spring 的方式。

小结

SpringBoot提供了一种快速开发Spring项目的方式，而不是对Spring功能上的增强。

Spring的缺点：

- 配置繁琐
- 依赖繁琐

SpringBoot功能：

- 自动配置
- 起步依赖：依赖传递
- 辅助功能

目录 Contents

SpringBoot 快速入门

案例：需求

搭建SpringBoot工程，定义HelloController.hello()方法，返回”Hello SpringBoot!”。

案例：实现步骤

① 创建Maven项目
② 导入SpringBoot起步依赖
③ 定义Controller
④ 编写引导类
⑤ 启动测试

小结

- SpringBoot在创建项目时，使用jar的打包方式。
- SpringBoot的引导类，是项目入口，运行main方法就可以启动项目。
- 使用SpringBoot和Spring构建的项目，业务代码编写方式完全一样。

目录 Contents

起步依赖原理分析

1） spring-boot-starter-parent

2） spring-boot-starter-web

SpringBoot 起步依赖原理分析

小结

- 在spring-boot-starter-parent中定义了各种技术的版本信息，组合了一套最优搭配的技术版本。
- 在各种starter中，定义了完成该功能需要的坐标合集，其中大部分版本信息来自于父工程。
- 我们的工程继承parent，引入starter后，通过依赖传递，就可以简单方便获得需要的jar包，并且不会存在版本冲突等问题。

目录 Contents

目录 Contents

SpringBoot 配置

配置文件分类

SpringBoot是基于约定的，所以很多配置都有默认值，但如果想使用自己的配置替换默认配置的话，就可以使用application.properties或者application.yml（application.yaml）进行配置。

- properties：

```
server.port=8080
```

- yml:

```
server:
  port: 8080
```

SpringBoot 配置

小结

- SpringBoot提供了2种配置文件类型：properteis和yml/yaml
- 默认配置文件名称：application
- 在同一级目录下优先级为：properties > yml > yaml

目录 Contents

YAML

YAML全称是 YAML Ain't Markup Language 。YAML是一种直观的能够被电脑识别的的数据数据序列化格式，并且容易被人类阅读，容易和脚本语言交互的，可以被支持YAML库的不同的编程语言程序导入，比如： C/C++, Ruby, Python, Java, Perl, C#, PHP等。YML文件是以数据为核心的，比传统的xml方式更加简洁。

YAML文件的扩展名可以使用.yml或者.yaml。

SpringBoot 配置

YAML

- properties:

```
server.port=8080
server.address=127.0.0.1
```

- xml:

```
<server>
    <port>8080</port>
    <address>127.0.0.1</address>
</server>
```

- yml:

```
server:
    port: 8080
    address: 127.0.0.1
```

简洁，以数据为核心

YAML：基本语法

- 大小写敏感
- 数据值前边必须有空格，作为分隔符
- 使用缩进表示层级关系
- 缩进时不允许使用Tab键，只允许使用空格（各个系统 Tab对应的 空格数目可能不同，导致层次混乱）。
- 缩进的空格数目不重要，只要相同层级的元素左侧对齐即可
- # 表示注释，从这个字符一直到行尾，都会被解析器忽略。

```yaml
server:
    port: 8080
    address: 127.0.0.1

name: abc
```

SpringBoot 配置

YAML：数据格式

- 对象(map)：键值对的集合。

```yaml
person:
  name: zhangsan
# 行内写法
person: {name: zhangsan}
```

- 数组：一组按次序排列的值

```yaml
address:
  - beijing
  - shanghai
# 行内写法
address: [beijing,shanghai]
```

- 纯量：单个的、不可再分的值

```yaml
msg1: 'hello \n world'  # 单引忽略转义字符
msg2: "hello \n world"  # 双引识别转义字符
```

YAML：参数引用

```yaml
name: lisi

person:
  name: ${name} # 引用上边定义的name值
```

YAML：小结

1） 配置文件类型

- properties：和以前一样
- yml/yaml：注意空格

2） yaml：简洁，以数据为核心

- 基本语法
  - 大小写敏感
  - 数据值前边必须有空格，作为分隔符
  - 使用空格缩进表示层级关系，相同缩进表示同一级
- 数据格式
  - 对象
  - 数组: 使用 “- ” 表示数组每个元素
  - 纯量
- 参数引用
  - ${key}

目录 Contents

SpringBoot 配置

读取配置内容

1） @Value

2） Environment

3） @ConfigurationProperties

目录 Contents

profile

我们在开发Spring Boot应用时，通常同一套程序会被安装到不同环境，比如：开发、测试、生产等。其中数据库地址、服务器端口等等配置都不同，如果每次打包时，都要修改配置文件，那么非常麻烦。profile功能就是来进行动态配置切换的。

1） profile配置方式
   - 多profile文件方式
   - yml多文档方式

2） profile激活方式
   - 配置文件
   - 虚拟机参数
   - 命令行参数

Profile-小结

1） profile是用来完成不同环境下，配置动态切换功能的。

2） profile配置方式

- 多profile文件方式：提供多个配置文件，每个代表一种环境。
  - application-dev.properties/yml 开发环境
  - application-test.properties/yml 测试环境
  - application-pro.properties/yml 生产环境
- yml多文档方式：
  - 在yml中使用 --- 分隔不同配置

3） profile激活方式

- 配置文件： 再配置文件中配置：spring.profiles.active=dev
- 虚拟机参数：在VM options 指定：-Dspring.profiles.active=dev
- 命令行参数：java –jar xxx.jar --spring.profiles.active=dev

目录 Contents

SpringBoot 配置

内部配置加载顺序

Springboot程序启动时，会从以下位置加载配置文件：

1. file:./config/：当前项目下的/config目录下
2. file:./ ：当前项目的根目录
3. classpath:/config/：classpath的/config目录
4. classpath:/ ：classpath的根目录

加载顺序为上文的排列顺序，高优先级配置的属性会生效

目录 Contents

SpringBoot 配置

外部配置加载顺序

通过官网查看外部属性加载顺序：

https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html

目录 Contents

SpringBoot 整合其他框架

案例：需求

SpringBoot整合Junit。

案例：实现步骤

① 搭建SpringBoot工程
② 引入starter-test起步依赖
③ 编写测试类
④ 添加测试相关注解
- @RunWith(SpringRunner.class)
- @SpringBootTest(classes = 启动类.class)

⑤ 编写测试方法

SpringBoot 整合其他框架

案例：需求

SpringBoot整合Redis。

案例：实现步骤

① 搭建SpringBoot工程
② 引入redis起步依赖
③ 配置redis相关属性
④ 注入RedisTemplate模板
⑤ 编写测试方法，测试

SpringBoot 整合其他框架

案例：需求

SpringBoot整合MyBatis。

案例：实现步骤

① 搭建SpringBoot工程
② 引入mybatis起步依赖，添加mysql驱动
③ 编写DataSource和MyBatis相关配置
④ 定义表和实体类
⑤ 编写dao和mapper文件/纯注解开发
⑥ 测试

SpringBoot 整合其他框架

整合dubbo

传智播客旗下高端IT教育品牌