Spring Boot 最流行的 16 条实践解读！

Jedrzejewski java大牛爱好者 4月2日

- 1、使用自定义BOM来维护第三方依赖
- 2、使用自动配置
- 3、使用Spring Initializr来开始一个新的Spring Boot项目
- 4、考虑为常见的组织问题创建自己的自动配置
- 5、正确设计代码目录结构
- 6、保持@Controller的简洁和专注
- 7、围绕业务功能构建@Service
- 8、使数据库独立于核心业务逻辑之外
- 9、保持业务逻辑不受Spring Boot代码的影响
- 10、推荐使用构造函数注入
- 11、熟悉并发模型
- 12、加强配置管理的外部化
- 13、提供全局异常处理
- 14、使用日志框架
- 15、测试你的代码
- 16、使用测试切片让测试更容易，并且更专注
- 总结

Spring Boot是最流行的用于开发微服务的Java框架。

在本文中，我将与你分享自2016年以来我在专业开发中使用Spring Boot所采用的最佳实践。

这些内容是基于我的个人经验和一些熟知的Spring Boot专家的文章。

在本文中，我将重点介绍Spring Boot特有的实践（大多数时候，也适用于Spring项目）。以下依次列出了最佳实践，排名不分先后。

> 欢迎胖友在文末留言，分享你的 **Spring Boot** 的最佳实践。

1、使用自定义BOM来维护第三方依赖

这条实践是我根据实际项目中的经历总结出的。

Spring Boot项目本身使用和集成了大量的开源项目，它帮助我们维护了这些第三方依赖。

但是也有一部分在实际项目使用中并没有包括进来，这就需要我们在项目中自己维护版本。

如果在一个大型的项目中，包括了很多未开发模块，那么维护起来就非常的繁琐。

怎么办呢？事实上，Spring IO Platform就是做的这个事情，它本身就是Spring Boot的子项目，同时维护了其他第三方开源库。

我们可以借鉴Spring IO Platform来编写自己的基础项目platform-bom，所有的业务模块项目应该以BOM的方式引入。

这样在升级第三方依赖时，就只需要升级这一个依赖的版本而已。

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>io.spring.platform</groupId>
            <artifactId>platform-bom</artifactId>
            <version>Cairo-SR3</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

2、使用自动配置

Spring Boot的一个主要特性是使用自动配置。这是Spring Boot的一部分，它可以简化你的代码并使之工作。

当在类路径上检测到特定的jar文件时，自动配置就会被激活。

使用它的最简单方法是依赖Spring Boot Starters。

因此，如果你想与Redis进行集成，你可以首先包括：

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

如果你想与MongoDB进行集成，需要这样：

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
</dependency>
```

借助于这些starters，这些繁琐的配置就可以很好地集成起来并协同工作，而且它们都是经过测试和验证的。

这非常有助于避免可怕的Jar地狱。

> https://dzone.com/articles/what-is-jar-hell

通过使用以下注解属性，可以从自动配置中排除某些配置类：

```
@EnableAutoConfiguration（exclude = {ClassNotToAutoconfigure.c
```

但只有在绝对必要时才应该这样做。

有关自动配置的官方文档可在此处找到：

> https://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-auto-configuration.html。

3、使用Spring Initializr来开始一个新的Spring Boot项目

> 这一条最佳实践来自Josh Long （Spring Advocate，@starbuxman）。

Spring Initializr 提供了一个超级简单的方法来创建一个新的Spring Boot项目，并根据你的需要来加载可能使用到的依赖。

> https://start.spring.io/

使用Initializr创建应用程序可确保你获得经过测试和验证的依赖项，这些依赖项适用于Spring自动配置。

你甚至可能会发现一些新的集成，但你可能并没有意识到这些。

4、考虑为常见的组织问题创建自己的自动配置

这一条也来自Josh Long（Spring Advocate，@starbuxman）——这个实践是针对高级用户的。

如果你在一个严重依赖Spring Boot的公司或团队中工作，并且有共同的问题需要解决，那么你可以创建自己的自动配置。

这项任务涉及较多工作，因此你需要考虑何时获益是值得投入的。

与多个略有不同的定制配置相比，维护单个自动配置更容易。

如果将这个提供Spring Boot配置以开源库的形式发布出去，那么将极大地简化数千个用户的配置工作。

5、正确设计代码目录结构

尽管允许你有很大的自由，但是有一些基本规则值得遵守来设计你的源代码结构。

避免使用默认包。

确保所有内容（包括你的入口点）都位于一个名称很好的包中，这样就可以避免与装配和组件扫描相关的意外情况；

将Application.java（应用的入口类）保留在顶级源代码目录中；

我建议将控制器和服务放在以功能为导向的模块中，但这是可选的。一些非常好的开发人员建议将所有控制器放在一起。不论怎样，坚持一种风格！

6、保持@Controller的简洁和专注

Controller应该非常简单。你可以在此处阅读有关GRASP中有关控制器模式部分的说明。

你希望控制器作为协调和委派的角色，而不是执行实际的业务逻辑。

以下是主要做法：

> https://en.wikipedia.org/wiki/GRASP_(object-oriented_design)#Controller

- 控制器应该是无状态的！默认情况下，控制器是单例，并且任何状态都可能导致大量问题；
- 控制器不应该执行业务逻辑，而是依赖委托；
- 控制器应该处理应用程序的HTTP层，这不应该传递给服务；
- 控制器应该围绕用例/业务能力来设计。

要深入这个内容，需要进一步地了解设计REST API的最佳实践。无论你是否想要使用Spring Boot，都是值得学习的。

7、围绕业务功能构建@Service

Service是Spring Boot的另一个核心概念。

我发现最好围绕业务功能/领域/用例（无论你怎么称呼都行）来构建服务。

在应用中设计名称类似AccountService, UserService, PaymentService这样的服务，比起像DatabaseService、ValidationService、CalculationService这样的会更合适一些。

你可以决定使用Controler和Service之间的一对一映射，那将是理想的情况。但这并不意味着，Service之间不能互相调用！

8、使数据库独立于核心业务逻辑之外

我之前还不确定如何在Spring Boot中最好地处理数据库交互。

在阅读了罗伯特·C·马丁的“Clear Architecture”之后，对我来说就清晰多了。

你希望你的数据库逻辑于服务分离出来。

理想情况下，你不希望服务知道它正在与哪个数据库通信，这需要一些抽象来封装对象的持久性。

> 罗伯特C.马丁强烈地说明，你的数据库是一个“细节”，这意味着不将你的应用程序与特定数据库耦合。
>
> 过去很少有人会切换数据库，我注意到，使用Spring Boot和现代微服务开发会让事情变得更快。

9、保持业务逻辑不受**Spring Boot**代码的影响

考虑到“Clear Architecture”的教训，你还应该保护你的业务逻辑。

将各种Spring Boot代码混合在一起是非常诱人的……不要这样做。

如果你能抵制诱惑，你将保持你的业务逻辑可重用。

部分服务通常成为库。如果不从代码中删除大量Spring注解，则更容易创建。

10、推荐使用构造函数注入

这一条实践来自Phil Webb（Spring Boot的项目负责人, @phillip_webb）。

保持业务逻辑免受Spring Boot代码侵入的一种方法是使用构造函数注入。

不仅是因为@Autowired注解在构造函数上是可选的，而且还可以在没有Spring的情况下轻松实例化bean。

11、熟悉并发模型

我写过的最受欢迎的文章之一是“介绍Spring Boot中的并发”。

我认为这样做的原因是这个领域经常被误解和忽视。如果使用不当，就会出现问题。

> https://www.e4developer.com/2018/03/30/introduction-to-concurrency-in-spring-boot/

在Spring Boot中，Controller和Service是默认是单例。

如果你不小心，这会引入可能的并发问题。

你通常也在处理有限的线程池,请熟悉这些概念。

如果你正在使用新的WebFlux风格的Spring Boot应用程序，我已经解释了它在“Spring’s WebFlux/Reactor Parallelism and Backpressure”中是如何工作的。

12、加强配置管理的外部化

这一点超出了Spring Boot，虽然这是人们开始创建多个类似服务时常见的问题……

你可以手动处理Spring应用程序的配置。

如果你正在处理多个Spring Boot应用程序，则需要使配置管理能力更加强大。

我推荐两种主要方法：

- 使用配置服务器，例如Spring Cloud Config；
- 将所有配置存储在环境变量中（可以基于git仓库进行配置）。

这些选项中的任何一个（第二个选项多一些）都要求你在DevOps更少工作量，但这在微服务领域是很常见的。

13、提供全局异常处理

你真的需要一种处理异常的一致方法。Spring Boot提供了两种主要方法：

- 你应该使用HandlerExceptionResolver定义全局异常处理策略；
- 你也可以在控制器上添加@ExceptionHandler注解，这在某些特定场景下使用可能会很有用。

这与Spring中的几乎相同，并且Baeldung有一篇关于REST与Spring的错误处理的详细文章，非常值得一读。

> https://www.baeldung.com/exception-handling-for-rest-with-spring

14、使用日志框架

你可能已经意识到这一点，但你应该使用Logger进行日志记录，而不是使用System.out.println()手动执行。

这很容易在Spring Boot中完成，几乎没有配置。只需获取该类的记录器实例：

```
Logger logger = LoggerFactory.getLogger(MyClass.class);
```

这很重要，因为它可以让你根据需要设置不同的日志记录级别。

15、测试你的代码

这不是Spring Boot特有的，但它需要提醒——测试你的代码！如果你没有编写测试，那么你将从一开始就编写遗留代码。

如果有其他人使用你的代码库，那边改变任何东西将会变得危险。

当你有多个服务相互依赖时，这甚至可能更具风险。

由于存在Spring Boot最佳实践，因此你应该考虑将Spring Cloud Contract用于你的消费者驱动契约，它将使你与其他服务的集成更容易使用。

16、使用测试切片让测试更容易，并且更专注

这一条实践来自Madhura Bhave（Spring 开发者, @madhurabhave23）。

使用Spring Boot测试代码可能很棘手——你需要初始化数据层，连接大量服务，模拟事物……实际上并不是那么难！答案是使用测试切片。

使用测试切片，你可以根据需要仅连接部分应用程序。

这可以为你节省大量时间，并确保你的测试不会与未使用的内容相关联。

来自spring.io的一篇名为Custom test slice with Spring test 1.4的博客文章解释了这种技术。

> https://spring.io/blog/2016/08/30/custom-test-slice-with-spring-boot-1-4

总结

感谢Spring Boot，编写基于Spring的微服务正变得前所未有的简单。

我希望通过这些最佳实践，你的实施过程不仅会变得很快，而且从长远来看也会更加强大和成功。祝你好运！

来源：http://t.cn/EJWZNra

设置星标不迷路

有你想看的精彩

当程序员成立了Hello World共和国……丨愚人节恶搞

程序员的愚人节 正确打开方式~

感谢您的阅读，如果你觉得我的公众号还不错，请多帮我推荐给你的朋友，多谢了。

Java大牛爱好者：每天一篇java技术文章，不定时java干货发送

欢迎长按（扫描）二维码关注：java大牛爱好者

福利：公众号点击“开课吧”，可以免费领取

【高级java架构师学习资料】-年薪百万不是梦！

【java学习电子书】-从入门到架构！

更多惊喜等你去发现~

万水千山总是情，点个 “**在看**” 行不行？