第1-2课：Spring Boot 2.0 都更新了什么（上）

2018 年 3 月 1 号 Spring Boot 2.0.0.RELEASE 正式发布，这是 Spring Boot 1.0 发布 4 年之后第一次重大修订，因此有多新功能和特性值得关注！在 Spring Boot 官方博客中我们了解到：Spring Boot 2.0 版本经历了 17 个月的开发，有 215 个不同的使用者提供了超过 6800 次的提交。

我们将 Spring Boot 2.0 更新的技术分为三类进行解读：

- 第一类，基础环境升级；
- 第二类，默认软件替换和优化；
- 第三类，引入新的技术。

基础环境升级

最低 JDK 8，支持 JDK 9，不再支持 Java 6 和 7

Spring Boot 2.0 要求 Java 8 作为最低版本，许多现有的 API 已更新，以利用 Java 8 的特性，例如，接口上的默认方法，函数回调以及新的 API，如 javax.time。如果你当前正在使用 Java 7 或更早版本，则在开发 Spring Boot 2.0 应用程序之前，需要升级你的 JDK。

Spring Boot 2.0 通过了在 JDK 9 下的测试，可以在 JDK 9 下正常运行，同时 Spring Boot 2.0 宣布不再支持 Java 6 和 7，据我了解国内绝大部分互联网公司的基本环境还在 JDK 7 或者 6 环境下运行，考虑升级 Spring Boot 2.0 的团队需要考虑这个因素。

依赖组件升级

Spring Boot 2.0 基于 Spring Framework 5 构建，本次 Spring Boot 的升级，同时也升级了部分其依赖的第三方组件，主要有以下几个：

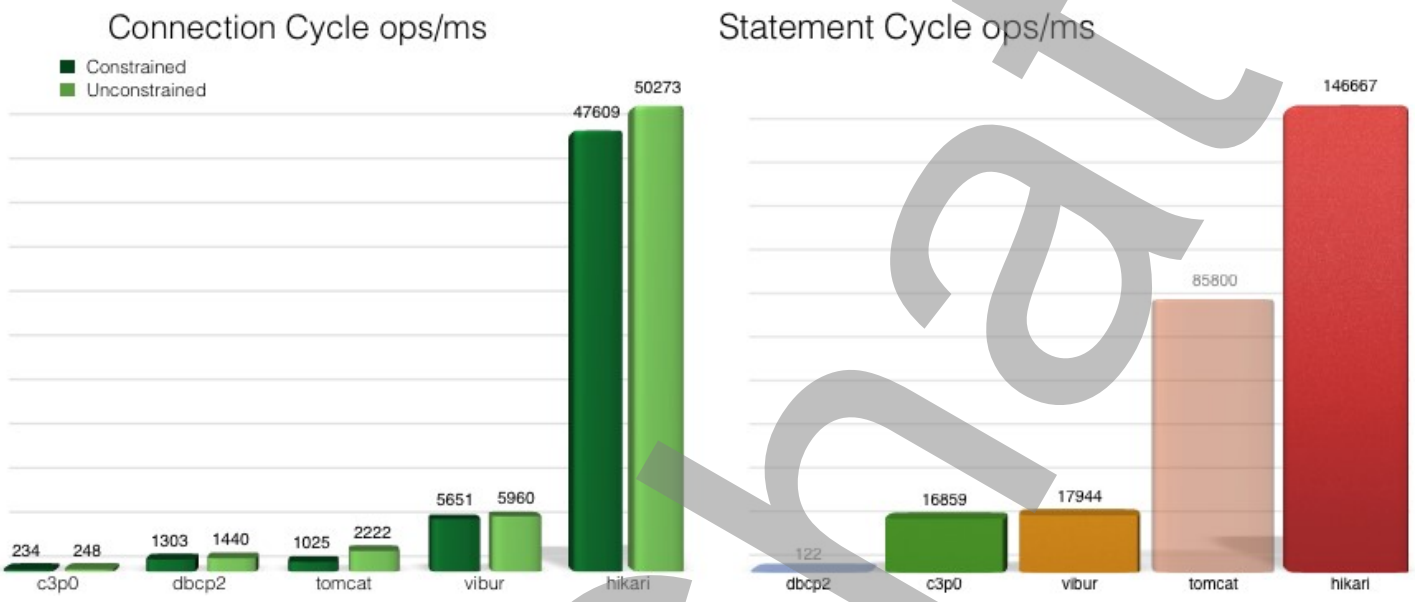
- Jetty 9.4，Jetty 是一个开源的 Servlet 容器，它为基于 Java 的 Web 内容，例如 JSP 和 Servlet 提供运行环境。Jetty 是使用 Java 语言编写的，它的 API 以一组 JAR 包的形式发布。
- Tomcat 8.5，Apache Tomcat 8.5.x 旨在取代 8.0.x，完全支持 Java 9。
- Flyway 5，Flyway 是独立于数据库的应用、管理并跟踪数据库变更的数据库版本管理工具。用通俗的话讲，Flyway 可以像 SVN 管理不同人的代码那样，管理不同人的 SQL 脚本，从而做到数据库同步。
- Hibernate 5.2，Hibernate 是一款非常流行的 ORM 框架。
- Gradle 3.4，Spring Boot 的 Gradle 插件在很大程度上已被重写，有了重大的改进。
- Thymeleaf 3.0，Thymeleaf 3 相对于 Thymeleaf 2 有非常大的性能提升。

默认软件替换和优化

HikariCP

默认连接池已从 Tomcat 切换到 HikariCP，HikariCP 是一个高性能的 JDBC 连接池，Hikari 是日语“光”的意思。

HikariCP 号称是 Java 业界最快的数据库连接池，官网提供了 c3p0、dbcp2、tomcat、vibur 和 Hikari 等数据连接池的性能对比。



关于性能为什么如此突出，官网给出的说明如下：

- 字节码精简：优化代码，直到编译后的字节码最少，这样，CPU 缓存可以加载更多的程序代码；
- 优化代理和拦截器：减少代码，例如 HikariCP 的 Statement proxy 只有 100 行代码；
- 自定义数组类型（FastStatementList）代替 ArrayList：避免每次 get() 调用都要进行 range check，避免调用 remove() 时从头到尾的扫描；
- 自定义集合类型（ConcurrentBag）：提高并发读写的效率；
- 其他针对 BoneCP 缺陷的优化，比如对于耗时超过一个 CPU 时间片的方法调用的研究。

Security

Spring Security 是 Spring 社区的一个顶级项目，也是 Spring Boot 官方推荐使用的 Security 框架。除了常规的 Authentication 和 Authorization 之外，Spring Security 还提供了诸如 ACLs、LDAP、JAAS、CAS 等高级特性以满足复杂场景下的安全需求。

没有使用 Spring Boot 之前集成起来相对比较麻烦，而 Spring Boot 中基于 Java 配置实现 Spring Security 功能。Spring Boot 2 极大地简化了默认的安全配置，并使添加定制安全变得简单。

Spring Boot 2.0 非常容易使用 Spring Security 5.0 保护响应式应用，当检测到 Spring Security 存在的时候会自动进行默认配置。

OAuth 2

OAuth 2.0 是 OAuth 协议的延续版本，但不向后兼容 OAuth 1.0，即完全废止了 OAuth1.0。OAuth 2.0 关注

客户端开发者的简易性。要么通过组织在资源拥有者和 HTTP 服务商之间的被批准的交互动作代表用户，要么允许第三方应用代表用户获得访问的权限。

OAuth 2 是一个授权框架，或称授权标准，它可以使第三方应用程序或客户端获得对 HTTP 服务上（如 Google、GitHub ）用户帐户信息的有限访问权限。OAuth 2 通过将用户身份验证委派给托管用户帐户的服务以及授权客户端访问用户帐户进行工作。

Spring Boot 2.0 将 Spring Security OAuth 项目迁移到 Spring Security。不再提供单独的依赖包，Spring Boot 2 通过 Spring Security 5 提供 OAuth 2.0 客户端支持。

Micrometer

Micrometer 是一款监控指标的度量类库，可以让你在没有供应商锁定的情况下对 JVM 的应用程序代码进行调整。

Spring Boot 2 增强了对 Micrometer 的集成，Spring Boot 2.0 不再提供自己的指标 API。依靠 micrometer.io 来满足所有应用程序监视需求。

Micrometer 包括尺寸指标的支持，当与尺寸监测系统配对时，尺寸指标可以有效访问特定的指定度量标准，并且可以在其尺寸范围内向下钻取。

指标可以输出到各种系统和开箱即用的 Spring Boot 2.0，为 Atlas、Datadog、Ganglia、Graphite、Influx、JMX、New Relic、Prometheus、SignalFx、StatsD 和 Wavefront 提供支持，另外还可以使用简单的内存中度量标准。

集成后提供 JVM 指标（包括 CPU、内存、线程和 GC）、Logback、Tomcat、Spring MVC & 提供 RestTemplate。

Redis 默认使用 Lettuce

Redis 方面默认引入了 Lettuce，替代了之前的 Jedis 作为底层的 Redis 连接方式。

Lettuce 是一个可伸缩的线程安全的 Redis 客户端，用于同步、异步和反应使用。多个线程可以共享同一个 RedisConnection，它利用优秀 Netty NIO 框架来高效地管理多个连接，支持先进的 Redis 功能，如 Sentinel、集群、流水线、自动重新连接和 Redis 数据模型。

国内使用 Jedis 的居多，看来以后要多研究 Lettuce 了。

配置属性绑定

在 Spring Boot 2.0 中，使用 Environment 绑定机制的 @ConfigurationProperties 数学已经完全彻底修改。借此机会收紧了绑定的规则，并修复了 Spring Boot 1.x 中的许多不一致之处。

新的 Binder API 也可以直接使用 @ConfigurationProperties 在代码中。例如，下面绑定 List 中的 PersonName 对象：

```
List<PersonName> people = Binder.get(environment)
    .bind("my.property", Bindable.listOf(PersonName.class))
    .orElseThrow(IllegalStateException::new);
```

配置源可以像这样在 YAML 中表示：

```
my:
  property:
  - first-name: Jane
    last-name: Doe
  - first-name: John
    last-name: Doe
```

转换器支持

Binding 使用了一个新的 ApplicationConversionService 类，它提供了一些额外有用的转化。最引人注目的是转换器的 Duration 类型和分隔字符串。

该 Duration 转换器允许在任一 ISO-8601 格式的持续时间，或是一个简单的字符串（如 10m，10 分钟）。现有的属性已更改为默认使用 Duration，该 @DurationUnit 注释通过设置如果没有指定所使用的单元确保向后兼容性。例如，Spring Boot 1.5 中需要秒数的属性现在必须 @DurationUnit(ChronoUnit.SECONDS) 确保一个简单的值，例如 10 实际使用的值为 10s。

分隔字符串转换允许你将简单绑定 String 到 Collection 或 Array 不必分割逗号。例如，LDAP `base-dn` 属性用 `@Delimiter(Delimiter.NONE)`，所以 LDAP DN（通常包含逗号）不会被错误解释。

Actuator 改进

在 Spring Boot 2.0 中 Actuator endpoints 有很大的改进，所有 HTTP Actuator endpoints 现在都在该 `/actuator` 路径下公开，并且生成的 JSON 有效负载得到了改进。

现在默认情况下不会暴露很多端点。如果你要从 Spring Boot 1.5 升级现有的应用，请务必查看迁移指南并特别注意该 management.endpoints.web.exposure.include 属性。

Spring Boot 2.0 改进了从许多端点返回的 JSON 有效负载。

现在许多端点都具有更精确地反映底层数据的 JSON。例如，`/actuator/conditions` 终端（`/autoconfig` 在 Spring Boot 1.5 中）现在有一个顶级 contexts 密钥来将结果分组 ApplicationContext。

测试

对 Spring Boot 2.0 中测试进行了一些补充和调整：

- @WebFluxTest 已添加新注释以支持 WebFlux 应用程序的“slice”测试。

- Converter 和 GenericConverter beans 现在自动扫描 @WebMvcTest 和 @WebFluxTest。
- @AutoConfigureWebTestClient 已经添加到 WebTestClient 供测试使用，这个注释会自动应用于 @WebFluxTest 测试。
- 增加了一个新的 ApplicationContextRunner 测试实用程序，可以很容易地测试你的自动配置，我们已将大部分内部测试套件移至此新模型。

其他

还有一些小的调整和改进：

- @ConditionalOnBean 现在在确定是否满足条件时使用逻辑 AND 而不是逻辑 OR。
- 无条件类现在包含在自动配置中。
- 该 spring CLI 应用程序现在包括 encodepassword 可用于创建 Spring Security 的兼容散列密码命令。
- 计划任务（即 @EnableScheduling）可以使用 scheduledtasks 执行器端点进行审查。
- 该 loggers 驱动器终端现在允许你重新设置一个日志的默认级别。
- Spring Session 用户现在可以通过 sessions 执行器端点查找和删除会话。
- 使用 spring-boot-starter-parent 现在基于 Maven 的应用程序 `-parameters` 默认使用标志。

引入新的技术

支持 HTTP/2

HTTP/2 是第二代的 HTTP 协议，Spring Boot 的 Web 容器选择中 Tomcat，Undertow 和 Jetty 均已支持 HTTP/2。

相比 HTTP/1.x，HTTP/2 在底层传输做了很大的改动和优化：

- HTTP/2 采用二进制格式传输数据，而非 HTTP/1.x 的文本格式。二进制格式在协议的解析和优化扩展上带来更多的优势和可能。
- HTTP/2 对消息头采用 HPACK 进行压缩传输，能够节省消息头占用的网络的流量；而 HTTP/1.x 每次请求，都会携带大量冗余头信息，浪费了很多带宽资源；头压缩能够很好的解决该问题。
- 多路复用，直白的说就是所有的请求都是通过一个 TCP 连接并发完成。HTTP/1.x 虽然通过 pipeline 也能并发请求，但是多个请求之间的响应会被阻塞的，所以 pipeline 至今也没有被普及应用，而 HTTP/2 做到了真正的并发请求。同时，流还支持优先级和流量控制。
- Server Push：服务端能够更快的把资源推送给客户端。例如，服务端可以主动把 JS 和 CSS 文件推送给客户端，而不需要客户端解析 HTML 再发送这些请求，当客户端需要的时候，它已经在客户端了。

嵌入式 Netty 服务器

由于 WebFlux 不依赖于 Servlet API，我们现在可以首次为 Netty 作为嵌入式服务器提供支持，该 spring-boot-starter-webflux 启动 POM 将拉取 Netty 4.1 和 Ractor Netty。

> 注意：你只能将 Netty 用作反应式服务器，不提供阻止 Servlet API 支持。

Kotlin

Spring Boot 2.0 现在包含对 Kotlin 1.2.x 的支持，并提供了 runApplication，一个使用 Kotlin 运行 Spring Boot 应用程序的方法。我们还公开和利用了 Kotlin 对其他 Spring 项目（如 Spring Framework，Spring Data 和 Reactor）已添加到其最近版本中的支持。

JOOQ 的支持

JOOQ 是基于 Java 访问关系型数据库的工具包。JOOQ 既吸取了传统 ORM 操作数据的简单性和安全性，又保留了原生 SQL 的灵活性，它更像是介于 ORMS 和 JDBC 的中间层。对于喜欢写 SQL 的码农来说，JOOQ 可以完全满足你控制欲，可以是用 Java 代码写出 SQL 的感觉来。

支持 Quartz

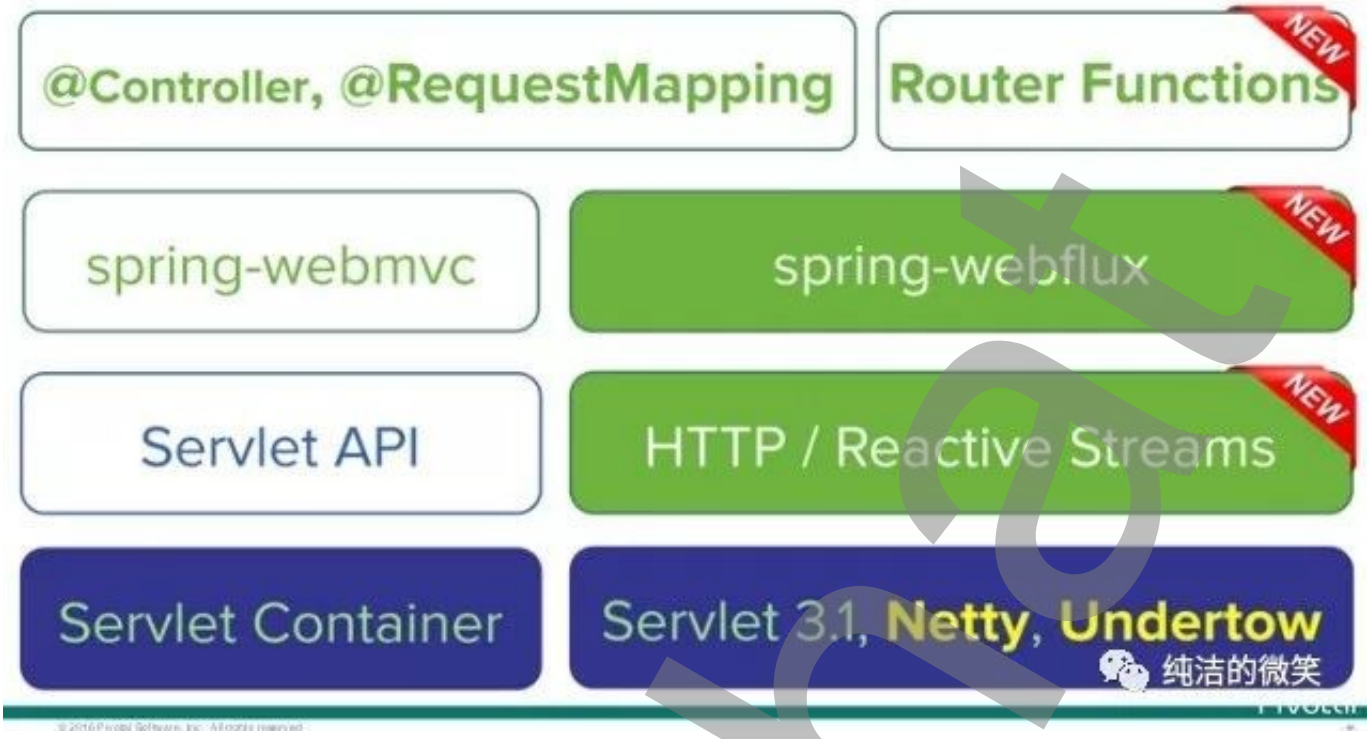
Spring Boot 1.0 并没有提供对 Quartz 的支持，之前出现了各种集成方案，Spring Boot2.0 给出了最简单的集成方式。

响应式编程

WebFlux 模块的名称是 spring-webflux，名称中的 Flux 来源于 Reactor 中的类 Flux。Spring WebFlux 有一个全新的非堵塞的函数式 Reactive Web 框架，可以用来构建异步的、非堵塞的、事件驱动的服务，在伸缩性方面表现非常好。

非阻塞的关键预期好处是能够以小的固定数量的线程和较少的内存进行扩展。在服务器端 WebFlux 支持两种不同的编程模型：

- 基于注解的 @Controller 和其他注解也支持 Spring MVC；
- Functional 、Java 8 Lambda 风格的路由和处理。



默认情况下，Spring Boot 2 使用 Netty WebFlux，因为 Netty 在异步非阻塞空间中被广泛使用，异步非阻塞连接可以节省更多的资源，提供更高的响应度。通过比较 Servlet 3.1 非阻塞 I/O 没有太多的使用，因为使用它的成本比较高，Spring WebFlux 打开了一条实用的通路。

使用 Spring WebFlux/WebFlux.fn 提供响应式 Web 编程支持，WebFlux 是一个全新的非堵塞的函数式 Reactive Web 框架，可以用来构建异步的、非堵塞的、事件驱动的服务，在伸缩性方面表现非常好，此功能来源于 Spring 5.0。

Spring Boot 2.0 也提供了对响应式编程的自动化配置，如 Reactive Spring Data、Reactive Spring Security 等。

最后，Spring Boot 2.0 的新技术还有一个有意思的彩蛋设计，我们会再下一讲介绍。同时，我会详细解释一下 Spring Boot 1.0 到 Spring Boot 2.0 API 上的一些变化，以及关于是否需要升级 Spring Boot 的个人建议。