Spring Boot 2: что нового?

[Автор оригинала: Dan Vega](https://medium.com/@therealdanvega/whats-new-in-spring-boot-2-c6c4cbd3b645)

*От переводчика: несмотря на то, что уже прошел год, вопросы о том что же нового дал нам 2-й Boot не заканчиваются. Писать такой контент с нуля — затея не самая умная. Поэтому решили перевести статью, которая нам кажется наиболее лаконичной и при этом достаточно полной.*

Релиз Spring Boot 2 состоялся в начале 2018 года, и всем не терпится скорее посмотреть на него в деле. Этот релиз стал кульминацией 17 месяцев работы и более 6800 коммитов от 215 разных людей. Есть много крутых функций, которые стоит обсудить, так что поговорим о том, что нового в Spring Boot 2.

В этой статье мы рассмотрим:

* Историю Spring Boot
* Что нового в Spring Boot
* Руководство по переходу на Spring Boot 2

История Spring Boot

Перед тем, как углубиться в нововведения, я бы хотел воспользоваться случаем и рассказать немного об истории Spring Boot. В посте, опубликованном в августе 2013, в своем блоге [Phil Webb объявил первый знаковый релиз проекта под названием Spring Boot](https://spring.io/blog/2013/08/06/spring-boot-simplifying-spring-for-everyone/).

*Spring Boot создан для того, чтобы сделать создание готовых к работе приложений и сервисов на Spring простым, без чрезмерных усилий. Spring Boot это субъективный взгляд на платформу Spring, который позволяет как новичкам так и опытным пользователям Spring-а найти все нужное. При помощи Boot-а можно создавать stand-alone приложения, запускаемые как ‘java -jar’ или распространяемые более традиционным WAR форматом.*

Спустя примерно 9 месяцев, в апреле 2014, состоялся релиз Spring Boot 1.0. С тех пор было множество второстепенных релизов и новых полезных функций.

[Spring Boot 1.1 (Июнь 2014)](https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-1.1-Release-Notes)

* spring-boot-starter-test
* Metrics & Health Endpoints
* Elastic Search, Apache Solr, Spring Social и Spring Integration авто-конфигурация
* Поддержка средств шаблонизации (Freemaker, Groovy, и Velocity)

[Spring Boot 1.2 (Март 2015)](https://github.com/spring-projects/spring-boot/wiki/spring-boot-1.2-release-notes)

* Servlet 3.1, Tomcat 8 & Jetty 9
* Spring 4.1
* Аннотация @SpringBootApplication
* Поддержка Email

[Spring Boot 1.3 (Декабрь 2016)](https://github.com/spring-projects/spring-boot/wiki/spring-boot-1.3-release-notes)

* Обновление Spring Framework до версии 4.2
* Обновление Spring Security до версии 4.0
* Developer Tools
* Caching авто-конфигурация
* Полностью исполняемые JAR’ы и поддержка сервисов

[Spring Boot 1.4 (Январь 2017)](https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-1.4-Release-Notes)

* Spring 4.3
* Hibernate 5
* Улучшения в механизмах тестирования
* Integration стартер
* Поддержка Couchbase и Neo4J

[Spring Boot 1.5 (Февраль 2017)](https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-1.5-Release-Notes)

* Loggers endpoint
* Поддержка Apache Kafka
* Расширенные механизмы для Cloud Foundry
* Поддержка LDAP
* Обновления тестирования

Что нового в Spring

Так что же нового в Spring Boot 2? Если говорить о самом большом нововведении, то это обновление Spring Framework до 5-ой версии. С тех пор, как в сентябре 2017 вышла Spring Framework 5, большинство разработчиков (как и я) ждали релиза Spring Boot 2. Spring Framework 5 имеет немалый список новых функций, но я бы хотел рассказать только о нескольких самых главных.

Что нового в Spring Framework 5

Поддержка версий Java 8+

Если вы и впредь хотите создавать приложения на Spring Framework, вам нужно работать с версией Java 8+. Вы, наверное, подумали, что это очень важное изменение для всех нас, но для команды Spring оно еще важнее. Это позволило обновить базу исходного кода до Java 8 со всеми его новыми фишками вроде лямбда-выражений или стримов. Это не только делает код более читабельным, но и улучшает производительность ядра платформы.

Поддержка Java 9

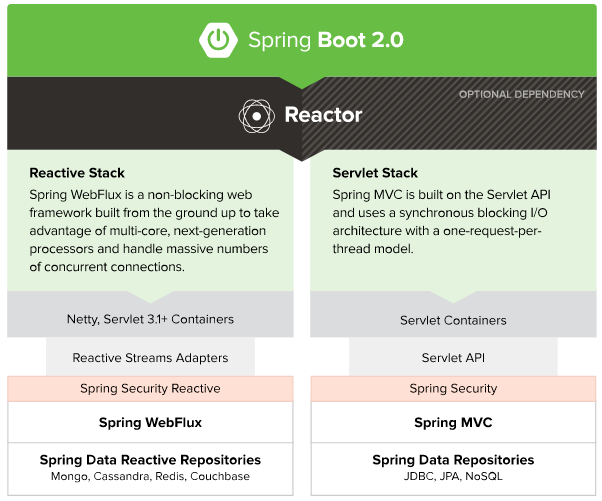
Если вы хотите использовать Java 9, вам нужно обновиться до Spring Framework 5 и также до Spring Boot 2. Я знаю, что многие еще не используют новейшие версии Java в продакшене, и это отличная возможность поэкспериментировать с новыми крутыми “игрушками”. Все должно работать без проблем при использовании стандартного classpath’а, но я читал о некоторых затруднениях при переходе на модули Java 9.

Spring MVC

Хотя Spring MVC и не в центре повествования в этой статье, стоит сказать, что было несколько приятных апгрейдов. Я не буду на них останавливаться, подробности можно посмотреть в [документации для Spring Framework 5](https://github.com/spring-projects/spring-framework/wiki/What%27s-New-in-Spring-Framework-5.x#spring-web-mvc).

Spring Webflux

Асинхронность потоков данных занимает центральное место в истории о Spring Framework 5. Это совершенно другой тип мышления, но, к счастью для нас, нет нужды переучиваться писать приложения совершенно новым способом. Spring WebFlux — абсолютно асинхронный и неблокирующий фреймворк, построенный с нуля, позволяющий справляться с большим количеством параллельных подключений. Хоть это и переворот в парадигме, начать будет не так уж сложно.



Поддержка Kotlin

Поддержка Kotlin была добавлена еще в [http://start.spring.io](http://start.spring.io/), но в Spring Framework 5 есть специализированная поддержка этого языка, привнесшая полезные функции, прочитать о них можно [здесь](https://github.com/spring-projects/spring-framework/wiki/What%27s-New-in-Spring-Framework-5.x#kotlin-support).

Улучшения в механизмах тестирования

Самое крупное изменение в системе тестирования — это полная поддержка [JUnit](http://junit.org/junit5/)5 Jupiter. Я еще расскажу об этом подробнее ниже, но когда вы запускаете новое приложение на Spring Boot 2, вы все еще используете по умолчанию JUnit 4, однако, переключиться на JUnit 5 — задача тривиальная.

Что нового в Spring Boot 2

Обновления сторонних библиотек

С каждым новым релизом Spring Boot у команды Spring возникает возможность обновить различные зависимости.

* Thymeleaf 3\*
* Jetty 9.4
* Tomcat 8.5
* Hibernate 5.2
* Flyway 5
* Gradle 4

\**В стартере Thymeleaf теперь есть встроенный thymeleaf-extras-java8time.*

Безопасность и Reactive Spring Data

С переходом на Spring WebFlux у Spring Data появилась поддержка приложений с асинхронными потоками данных. В настоящее время у Cassandra, MongoDB, Couchbase и Redis есть поддержка асинхронного API. В Spring Boot есть POM-стартеры для всех них, благодаря которым начать работать с ними совсем просто.

Также появилась возможность использовать Spring Security 5.0 в наших реактивных приложениях.

Actuator

Spring Boot Actuator не представляет из себя ничего нового, но он был переписан с нуля. Если вы еще не знакомы с Actuator, вот что он делает: он автоматически выводит endpoint’ы для получения информации о статусе приложения. Actuator в Spring Boot 1.x был написан поверх сервлетов, а с новыми реактивными подходами команде Spring нужно было решение которое поддерживает и старый и реактивный подходы. Помимо этого, в Actuator были внесены следующие изменения:

* Редизайн для поддержки сервлетов и реактивщины
* Статус и детальный health-check стали разделены
* Упрощенная модель безопасности
* Переход на Micrometer (это как SLF4J, только для метрик)
* Улучшенная структура данных JSON
* Упрощенный процесс создания кастомных endpoint-ов.
* [Endpoint](https://habr.com/en/users/endpoint/)
* @WebEndpoint
* @JmxEndpoint

Вероятно, тут могут возникнуть трудности в апгрейде из-за изменений в модели безопасности, о которых мы поговорим позже. По умолчанию все web-endpoint’ы доступны по пути /actuator с URL, имеющим вид /actuator/{id}. Путь /actuator можно изменить в настройках management.endpoints.web.base-path.

Существует подробный отдельный [блок документации для Spring Boot Actuator Web API Endpoints](https://docs.spring.io/spring-boot/docs/2.0.x/actuator-api/html/), с него стоит начать знакомство с этим инструментом.

Плагин Gradle

Мне всегда очень нравился Gradle, и я просто в восторге, что команда решила переписать плагин для Gradle.  
Spring Boot Gradle плагин позволяет реализовать поддержку Spring Boot в Gradle и дает возможность упаковывать исполняемые jar или war архивы, запускать приложения на Spring Boot и управлять зависимостями через spring-boot-dependencies. Для плагина Gradle в Spring Boot необходима версия Gradle 4.0 и старше.

**Начало работы**

Чтобы начать работать с плагином, нужно подгрузить его к проекту.

buildscript {

repositories {

maven { url 'https://repo.spring.io/libs-milestone' }

}

dependencies {

classpath 'org.springframework.boot:spring-boot-gradle-plugin:2.0.0.RC1'

}

}

apply plugin: 'org.springframework.boot'

**Создание исполняемых jar и war-архивов**

Таска bootRepackage была заменена на bootJar и bootWar для создания соответственно jar и war-файлов. Обе задачи обладают более широким функционалом по сравнению со стандартной аналогичной командой Gradle, обеспечивая доступ ко всем настройкам и логике.

**Управление зависимостями**

Плагин Gradle в Spring Boot больше не тянет плагин для управления зависимостями автоматически. Вместо этого Gradle плагин Spring Boot-а видит, что плагин управления зависимости подключен, и импортирует правильную версию BOM. Это позволяет лучше контролировать, где и каким образом конфигурируется управление зависимостями. Для большинства приложений будет достаточно добавить плагин управления зависимостями:

apply plugin: 'io.spring.dependency-management'

У [плагина Gradle есть своя документация](https://docs.spring.io/spring-boot/docs/2.0.x/gradle-plugin/reference/html/), которая содержит очень полезную информацию. Я рекомендую всем, кто хочет начать работать с Gradle, перейти по этой ссылке.

Упрощенная система безопасности

Одной из главных целей в Spring Boot 2.x было упростить конфигурацию системы безопасности и сделать так, чтобы можно было легко добавлять настраиваемые параметры безопасности. По умолчанию все данные защищены, включая статические ресурсы и эндпоинты Actuator-ов. Если Spring Security находится в classpath-е, Spring Boot добавит аннотацию @EnableWebSecurity, а какой конкретно механизм аутентификации будет использоваться определит механизм согласования содержимого (content-negotiation) из Spring Security.

При настройке пользователем своих правил безопасности стандартная конфигурация системы безопасности Spring Boot перестанет действовать. На этом этапе пользователю будет необходимо как можно точно прописать все правила безопасности. Это означает, что все настройки безопасности собраны в одном месте и проблем с порядком обработки команд при существующем WebSecurityConfigurerAdapters не возникает.

Пример индивидуальных настроек безопасности:

http

.authorizeRequests()

// 1

.requestMatchers(EndpointRequest.to("status", "info"))

.permitAll()

// 2

.requestMatchers(EndpointRequest.toAnyEndpoint())

.hasRole("ACTUATOR")

// 3

.requestMatchers(StaticResourceRequest.toCommonLocations())

.permitAll()

// 4

.antMatchers("/\*\*")

.hasRole("USER")

.and()

... // additional configuration

1. /status и /info не требуют авторизации.
2. Все прочие actuator защищены с помощью роли ACTUATOR.
3. Местонахождение общих статических ресурсов доступно всем.
4. Все прочие endpoint’ы приложения защищены ролью USER.

Actuator Security

С учетом того, что endpoint-ы actuator-а зафиксированы, необходимо включить или исключить endpoint’ы, в зависимости от того какие вам нужны. Вот настройки, которые это контролируют, с выпуска 1.x они изменились.

# ENDPOINTS WEB CONFIGURATION (WebEndpointProperties)

management.endpoints.web.exposure.include=info,health # Endpoint IDs that should be included or '\*' for all.

management.endpoints.web.exposure.exclude= # Endpoint IDs that should be excluded.

management.endpoints.web.base-path=/actuator # Base path for Web endpoints. Relative to server.servlet.context

path or management.server.servlet.context-path if management.server.port is configured.

management.endpoints.web.path-mapping= # Mapping between endpoint IDs and the path that should expose them.

Если вы не уверены, как настраивать эти параметры, добавьте в закладки [документацию по общим настройкам приложения](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/common-application-properties.html).

Поддержка HTTP/2

Невероятно, но релиз спецификации HTTP 1.1 был аж в 1996 году. Полагаю, можно не напоминать, что современная сеть отличается по всем параметрам. Если вы хотите реализовать поддержку HTTP/2 в приложении на Spring MVC или WebFlux, используйте следующий параметр.

server.http2.enabled=true

Поддержка HTTP/2 зависит от выбранного веб-сервера и среды приложения, так как этот протокол не поддерживается в JDK8 из коробки. Подробнее об этом написано [в документации](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/howto-embedded-web-servers.html#howto-configure-http2).

Конфигурационные параметры

В Spring Boot 1.x был поддержан так называемый relaxed binding, другими словами вы могли определить имя параметра многими способами (camel-case, нижнее подчеркивание, дефис) при этом в конечном итоге значение присваивалось одному и тому же свойству.

Relaxed binding работает все так же, а способ чтения переменных в вашем коде изменился:



[24. Externalized Configuration](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/boot-features-external-config.html#boot-features-external-config-relaxed-binding)

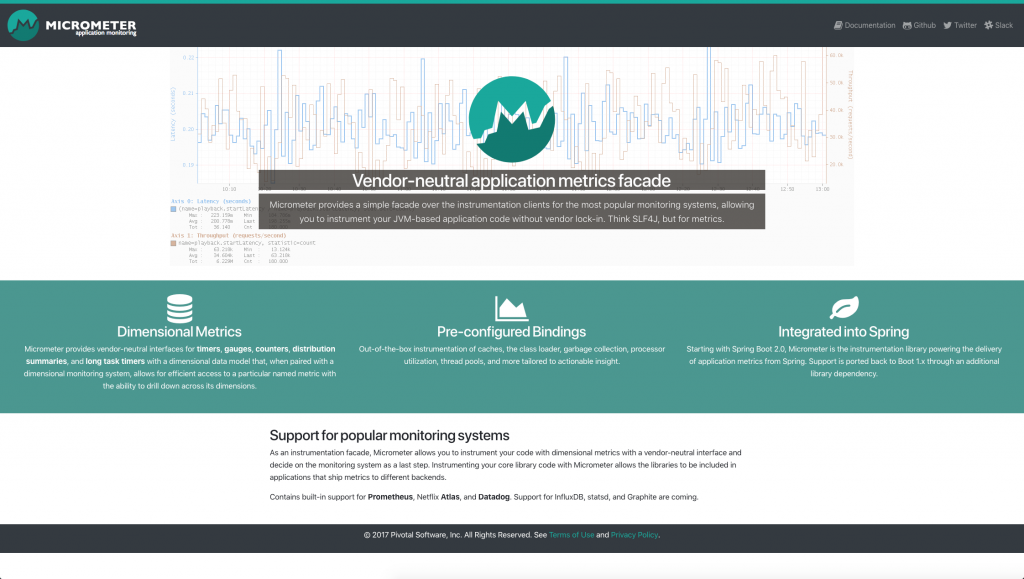
Spring Boot lets you externalize your configuration so that you can work with the same application code in different…docs.spring.io

Метрика

Собственная метрика Spring Boot была заменена на Micrometer. Он разрабатывается командой Pivotal и быстро адаптируется к проектам Pivotal.

Spring Boot Actuator предоставляет автоконфигурацию для [Micrometer](https://micrometer.io/) — интерфейса метрики приложения, поддерживающий множество видов мониторинга, включая:

* [Atlas](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/production-ready-metrics.html#production-ready-metrics-export-atlas)
* [Datadog](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/production-ready-metrics.html#production-ready-metrics-export-datadog)
* [Ganglia](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/production-ready-metrics.html#production-ready-metrics-export-ganglia)
* [Graphite](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/production-ready-metrics.html#production-ready-metrics-export-graphite)
* [Influx](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/production-ready-metrics.html#production-ready-metrics-export-influx)
* [JMX](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/production-ready-metrics.html#production-ready-metrics-export-jmx)
* [New Relic](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/production-ready-metrics.html#production-ready-metrics-export-newrelic)
* [Prometheus](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/production-ready-metrics.html#production-ready-metrics-export-prometheus)
* [SignalFx](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/production-ready-metrics.html#production-ready-metrics-export-signalfx)
* [Simple (in-memory)](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/production-ready-metrics.html#production-ready-metrics-export-simple)
* [StatsD](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/production-ready-metrics.html#production-ready-metrics-export-statsd)
* [Wavefront](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/production-ready-metrics.html#production-ready-metrics-export-wavefront)



Больше информации о Micrometer по ссылке <https://micrometer.io/>.

Апгрейд из-за этого может оказаться болезненным для тех, кто логирует тонны кастомных метрик.

Quartz Scheduler

В Spring Boot 2 есть поддержка библиотеки исполнения задач по расписанию Quartz, которую легко добавить при помощи стартера spring-boot-starter-quartz. Поддерживается работа на оперативной памяти и на JDBC.

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-quartz</artifactId>

</dependency>

HikariCP Connection Pool

Стандартный пул соединений изменился с Tomcat на HikariCP. Если вы использовали spring.datasource.type, чтобы заставить Hikari работать в приложении Tomcat, можете его убрать. Аналогично, если вы хотите продолжить пользоваться пулом Tomcat, просто добавьте к вашей конфигурации следующее:

spring.datasource.type=org.apache.tomcat.jdbc.pool.DataSource

Инструменты разработчика

По умолчанию при каждом перезапуске приложения записывается отчет о различии конфигураций. Другими словами, отчет показывает изменения в авто-конфигурации приложения, возникающие по мере введения вами таких изменений, как удаление/добавление бинов и настройка параметров конфигурации.

Чтобы отключить запись этого отчета, настройте следующий параметр:

spring.devtools.restart.log-condition-evaluation-delta=false

Поддержка Kotlin

Ранее в этой статье упоминалось об официальной поддержке Kotlin. Также есть [специализированный блок документации по Kotlin](https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/boot-features-kotlin.html).

JUnit 5

Как уже говорилось выше, в приложениях на Spring Boot по умолчанию все еще используется JUnit 4. Если вы хотите перейти на JUnit 5, нужно исключить JUnit 4 из spring-boot-starter-test и добавить необходимые зависимости. Также понадобятся плагины из списка ниже.

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

<exclusions>

<exclusion>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-api</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<dependencies>

<dependency>

<groupId>org.junit.platform</groupId>

<artifactId>junit-platform-surefire-provider</artifactId>

<version>${junit-platform.version}</version>

</dependency>

</dependencies>

</plugin>

</plugins>

</build>

Руководство по переходу на Spring Boot 2

Думаю, можно не говорить о настолько очевидном факте, что при таком крупном релизе просто сменить номер версии в продакшене — не лучший способ апгрейда. В первую очередь я бы посоветовал прочитать [Spring Boot 2.0 Migration Guide](https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-2.0-Migration-Guide). Лично я нашел решение большинства своих проблем в упрощенной модели безопасности и изменениях параметров. В этом гайде есть замечательные советы по переносу файлов параметров.

*В Spring Boot 2.0 многие параметры конфигурации были переименованы/удалены, и разработчикам необходимо обновить application.properties/application.yml с учетом этих изменений. Для облегчения этой задачи в Spring Boot реализована поставка нового модуля spring-boot-properties-migrator. Будучи добавленным к вашему проекту в качестве зависимости, он не только будет анализировать среду приложения и выводить результаты диагностики при запуске, но также временно мигрирует параметры в рантайме. Это необходимое действие при миграции приложения:*

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-properties-migrator</artifactId>

</dependency>

Не знаю, что вы об этом думаете, а я точно начну возиться с Spring Boot 2 и работать над миграцией кода на эту версию, но обычно при любом основном релизе я жду выпуска следующей минорной версии. Это касается не только Spring, но и других брендов от Apple до Pivotal и даже Angry Birds!