

УДК 004.4

## МОНИТОРИНГ ПРИЛОЖЕНИЙ С ПОМОЩЬЮ SPRING BOOT ACTUATOR

Е.П. БОБРОВИЧ

(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)

В статье представлен способ мониторинга и контроля Spring Boot приложений с помощью Actuator. Описана функциональность данного инструмента, настройки и возможности интеграции со внешними системами мониторинга приложений.

**Введение.** После того как приложение разработано и развернуто в рабочем окружении, очень важно следить за его работоспособностью. Корпоративные приложения меняются и становятся многоуровневыми, распределенными между разными серверами или даже континентами, переходя в облака. Поэтому такие сложные распределенные приложения требуют контроля, так как в некоторых компаниях являются основой бизнеса. Мониторинг производительности приложений решает задачи контроля, управления доступностью и непосредственно производительностью приложений

**Основная часть.** Spring Boot Actuator является подпроектом в проекте Spring Boot. Модуль Spring Boot Actuator помогает контролировать и управлять Spring Boot приложением, предоставляя готовые к работе функции, такие как проверка работоспособности, аудит, сбор метрик, трассировка HTTP и т.д.

Actuator также интегрируется с внешними системами мониторинга приложений, такими как Prometheus, Graphite, DataDog, Influx, Wavefront, New Relic и многими другими. Эти системы предоставляют отличные информационные панели, графики, аналитику и аварийные сигналы, которые помогут вам контролировать и управлять вашим приложением из единого интерфейса. Actuator использует Micrometer, фасад метрик приложений, для интеграции с этими внешними системами мониторинга приложений. Это позволяет очень легко подключить любую систему мониторинга приложений с минимальными настройками.

Чтобы включить Actuator, просто нужно добавить зависимость к менеджеру пакетов Spring Boot приложения. Примеры зависимостей для Maven и Gradle представлены в листингах 1 и 2 соответственно.

Листинг 1 – Добавление зависимости в проект Maven

```
1 <dependencies>
2   <dependency>
3     <groupId>org.springframework.boot</groupId>
4     <artifactId>spring-boot-starter-actuator</artifactId>
5   </dependency>
6 </dependencies>
```

Листинг 2 – Добавление зависимости в проект Gradle

```
1 dependencies {
2   compile("org.springframework.boot:spring-boot-starter-actuator")
3 }
```

Actuator создает несколько так называемых конечных точек, которые могут быть доступны через HTTP или JMX, чтобы вы могли отслеживать и взаимодействовать с вашим приложением. Например, существует конечная точка /health, которая предоставляет базовую информацию о состоянии приложения. Конечная точка /metrics показывает несколько полезных метрических данных, таких как используемая память JVM, загрузка ЦП системы, открытые файлы и многое другое.

По умолчанию все конечные точки открываются через JMX, но через HTTP предоставляются только /health и /info. Для доступа ко всем конечным точкам по HTTP следует в файле application.properties добавить свойство management.endpoints.web.exposure.include = \*. Если необходимо добавить доступ только к некоторым конечным точкам, то следует в значении свойства перечислить id этих конечных точек.

Список доступных конечных точек доступных по HTTP можно получить по базовой конечной точке /actuator. Перечень конечных точек представлен в таблице.

Конечная точка /health показывает только простое состояние UP или DOWN. Чтобы получить полную информацию, включая состояние каждого индикатора работоспособности, который был проверен в процессе проверки работоспособности, добавьте в файл application.properties свойство management.endpoint.health.show-details = always. После этого /health отображает больше информации

включая детали `DiskSpaceHealthIndicator`, который запускается как часть процесса проверки работоспособности приложения. Если к вашему приложению подключена база данных, то `/health` покажет текущее состояние подключения к ней. Вы также можете создать собственный индикатор работоспособности, реализовав интерфейс `HealthIndicator` или расширив класс `AbstractHealthIndicator`.

Таблица. – Конечные точки Spring Boot Actuator и их описание

ID конечной точки	Описание
<code>auditevents</code>	Предоставляет информацию о событиях аудита для текущего приложения
<code>beans</code>	Отображает полный список всех Spring-бинов в приложении
<code>caches</code>	Информация о кэше
<code>conditions</code>	Показывает условия, которые были вычислены для классов конфигурации и автоконфигурации, и причины, по которым они соответствовали или не соответствовали
<code>configprops</code>	Отображает список всех <code>@ConfigurationProperties</code>
<code>env</code>	Отображает свойства из <code>ConfigurableEnvironment</code>
<code>flyway</code>	Показывает миграции баз данных Flyway, которые были применены
<code>health</code>	Показывает сведения о работоспособности приложения
<code>httptrace</code>	Отображает информацию трассировки HTTP (по умолчанию последние 100 HTTP запросов-ответов)
<code>info</code>	Отображает дополнительную информацию о приложении
<code>integrationgraph</code>	Граф Spring Integration
<code>loggers</code>	Отображает и позволяет изменить конфигурацию логгеров в приложении
<code>liquibase</code>	Показывает примененные миграции базы данных Liquibase
<code>metrics</code>	Показывает информацию о метриках для текущего приложения
<code>mappings</code>	Отображает список всех путей <code>@RequestMapping</code>
<code>scheduledtasks</code>	Отображает запланированные задачи
<code>sessions</code>	Позволяет извлекать и удалять пользовательские сессии из хранилищ, поддерживаемых Spring Session. Недоступно при использовании Spring Session для реактивных веб-приложений
<code>shutdown</code>	Позволяет приложению корректно завершить работу
<code>threaddump</code>	Отображает информацию о потоках

Конечная точка `/metrics` перечисляет все метрики, которые вы можете отслеживать. Чтобы получить подробную информацию об отдельной метрике, вам нужно передать имя метрики в URL, как этот `http://localhost:8080/actuator/metrics/{MetricName}`. В ответ будут получены детали метрики в формате JSON.

Конечная точка `/loggers` отображает список всех настроенных в приложении логгеров с соответствующими уровнями логирования. Вы также можете просмотреть сведения об отдельном логгере, передав его имя в URL-адресе, например: `http://localhost:8080/actuator/loggers/{name}`. Конечная точка `/loggers` также позволяет во время выполнения приложения изменять уровень логирования указанного логера. Эта функциональность действительно будет полезна в тех случаях, когда ваше приложение сталкивается с проблемами в работе, и вы хотите на некоторое время включить логгирование DEBUG, чтобы получить более подробную информацию о проблеме.

Конечная точка `/info` отображает произвольную информацию о вашем приложении. Actuator получает информацию о сборке из файла `META-INF/build-info.properties`, информацию о Git из файла `git.properties`. Он также отображает любую информацию, доступную в свойствах среды с ключом `info`. Вы можете добавить свойства с ключом `info` в файле `application.properties` как в листинге 3.

Листинг 3 – Добавление зависимости в проект Gradle

```

1 info.app.name=@project.name@
2 info.app.description=@project.description@
3 info.app.version=@project.version@
4 info.app.java.version=@java.version@

```

Конечные точки Spring Boot Actuator рекомендуется защищать от неавторизованного доступа. Если в вашем приложении присутствует Spring Security, то эти конечные точки будут защищены по умолчанию. Так же можно задать свои настройки безопасности для них.

Spring Boot Actuator позволяет реализовывать свои пользовательские контрольные точки. Для получения конечной точки необходим bean, который можно создать с помощью аннотации `@Component`. Чтобы обращаться к конечной точке следует в аннотации `@Endpoint` задать параметр `id`, который будет

отвечать за путь /actuator/{id}. Аннотации @ReadOperation, @WriteOperation, @DeleteOperation обрабатывают HTTP GET, POST и DELETE соответственно и выполняют методы к которым относятся.

**Заключение.** Приведено описание мощного инструмента для контроля, мониторинга и управления приложениями Spring Boot Actuator. Данный подпроект Spring Boot имеет возможность гибкой настройки, кастомизации и создания пользовательских конечных точек. Actuator можно использовать с внешними системами мониторинга, что позволяет пользователю работать со всеми метриками и параметрами через единый интерфейс.

#### ЛИТЕРАТУРА

1. Spring Boot Actuator Web API Documentation [Электронный ресурс] / Spring Docs – 2019 – Режим доступа: <https://docs.spring.io/spring-boot/docs/current/actuator-api/html/> – Дата доступа: 26.09.2019.
2. Spring Boot Actuator: Production-ready features - Endpoints [Электронный ресурс] / Spring Docs – 2019. – Режим доступа: <https://docs.spring.io/spring-boot/docs/current/reference/html/production-ready-endpoints.html> – Дата доступа: 26.09.2019.