Spring Boot Actuator Web API Documentation

Andy Wilkinson

Table of Contents

1. Overview 2
   1.1. URLs 2
   1.2. Timestamps 2
2. Audit Events (`auditevents`) 3
   2.1. Retrieving Audit Events 3
      2.1.1. Query Parameters 3
      2.1.2. Response Structure 3
3. Beans (`beans`) 5
   3.1. Retrieving the Beans 5
      3.1.1. Response Structure 6
4. Conditions Evaluation Report (`conditions`) 8
   4.1. Retrieving the Report 8
      4.1.1. Response Structure 9
5. Configuration Properties (`configprops`) 11
   5.1. Retrieving the `@ConfigurationProperties` Bean 11
      5.1.1. Response Structure 12
6. Environment (`env`) 13
   6.1. Retrieving the Entire Environment 13
      6.1.1. Response Structure 14
   6.2. Retrieving a Single Property 15
      6.2.1. Response Structure 15
7. Flyway (`flyway`) 17
   7.1. Retrieving the Migrations 17
      7.1.1. Response Structure 17
8. Health (`health`) 19
   8.1. Retrieving the Health 19
      8.1.1. Response Structure 19
9. Heap Dump (`heapdump`) 21
   9.1. Retrieving the Heap Dump 21
10. Info (`info`) 22
   10.1. Retrieving the Info 22
      10.1.1. Response Structure 22
         `build` Response Structure 22
         `git` Response Structure 23
11. Liquibase (`liquibase`) 24
   11.1. Retrieving the Changes 24
      11.1.1. Response Structure 24
12. Log File (`logfile`) 26

12.1. Retrieving the Log File 26
12.2. Retrieving Part of the Log File 28
13. Loggers (loggers) 29
13.1. Retrieving All Loggers 29
13.1.1. Response Structure 29
13.2. Retrieving a Single Logger 30
13.2.1. Response Structure 30
13.3. Setting a Log Level 30
13.3.1. Request Structure 30
13.4. Clearing a Log Level 31
14. Metrics (metrics) 32
14.1. Retrieving Metric Names 32
14.1.1. Response Structure 32
14.2. Retrieving a Metric 32
14.2.1. Query Parameters 33
14.2.2. Response structure 33
14.3. Drilling Down 34
15. Prometheus (prometheus) 35
15.1. Retrieving the Metrics 35
16. Scheduled Tasks (scheduledtasks) 37
16.1. Retrieving the Scheduled Tasks 37
16.1.1. Response Structure 37
17. Sessions (sessions) 39
17.1. Retrieving Sessions 39
17.1.1. Query Parameters 40
17.1.2. Response Structure 40
17.2. Retrieving a Single Session 40
17.2.1. Response Structure 41
17.3. Deleting a Session 41
18. Shutdown (shutdown) 42
18.1. Shutting Down the Application 42
18.1.1. Response Structure 42
19. Thread Dump (threaddump) 43
19.1. Retrieving the Thread Dump 43
19.1.1. Response Structure 47

This API documentation describes Spring Boot Actuators web endpoints.

Chapter 1. Overview

Before you proceed, you should read the following topics:

- URLs
- Timestamps

1.1. URLs

By default, all web endpoints are available beneath the path `/actuator` with URLs of the form `/actuator/{id}`. The `/actuator` base path can be configured by using the `management.endpoints.web.base-path` property, as shown in the following example:

```
management.endpoints.web.base-path=/manage
```

The preceding `application.properties` example changes the form of the endpoint URLs from `/actuator/{id}` to `/manage/{id}`. For example, the URL `info` endpoint would become `/manage/info`.

1.2. Timestamps

All timestamps that are consumed by the endpoints, either as query parameters or in the request body, must be formatted as an offset date and time as specified in ISO 8601.

Chapter 2. Audit Events (auditevents)

The auditevents endpoint provides information about the application's audit events.

2.1. Retrieving Audit Events

To retrieve the audit events, make a GET request to /actuator/auditevents, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/auditevents?principal=alice&after=2017-11-
30T04%3A54%3A40.25Z&type=logout' -i
```

The preceding example retrieves logout events for the principal, alice, that occurred after 09:37 on 7 November 2017 in the UTC timezone. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 121

{
  "events" : [ {
    "timestamp" : "2017-11-30T04:54:40+0000",
    "principal" : "alice",
    "type" : "logout"
  } ]
}
```

2.1.1. Query Parameters

The endpoint uses query parameters to limit the events that it returns. The following table shows the supported query parameters:

Parameter	Description
after	Restricts the events to those that occurred after the given time. Required.
principal	Restricts the events to those with the given principal. Optional.
type	Restricts the events to those with the given type. Optional.

The after parameter is required. You can also use one or both of the principal and type parameters to further limit the results.

2.1.2. Response Structure

The response contains details of all of the audit events that matched the query. The following table

describes the structure of the response:

Path	Type	Description
`events`	`Array`	An array of audit events.
`events.[].timestamp`	`String`	The timestamp of when the event occurred.
`events.[].principal`	`String`	The principal that triggered the event.
`events.[].type`	`String`	The type of the event.

Chapter 3. Beans (beans)

The beans endpoint provides information about the application's beans.

3.1. Retrieving the Beans

To retrieve the beans, make a GET request to /actuator/beans, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/beans' -i
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 1119

{
  "contextId" : "application:-1",
  "beans" : {
    "defaultServletHandlerMapping" : {
      "aliases" : [ ],
      "scope" : "singleton",
      "type" :
"org.springframework.web.servlet.config.annotation.WebMvcConfigurationSupport$EmptyHan
dlerMapping",
      "resource" :
"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoConfiguration$EnableWebM
vcConfiguration",
      "dependencies" : [ ]
    },

"org.springframework.boot.autoconfigure.web.servlet.DispatcherServletAutoConfiguration
" : {
      "aliases" : [ ],
      "scope" : "singleton",
      "type" :
"org.springframework.boot.autoconfigure.web.servlet.DispatcherServletAutoConfiguration
$$EnhancerBySpringCGLIB$$f0ea3abd",
      "resource" : null,
      "dependencies" : [ ]
    },

"org.springframework.boot.autoconfigure.context.PropertyPlaceholderAutoConfiguration"
: {
      "aliases" : [ ],
      "scope" : "singleton",
      "type" :
"org.springframework.boot.autoconfigure.context.PropertyPlaceholderAutoConfiguration$$
EnhancerBySpringCGLIB$$33e5839",
      "resource" : null,
      "dependencies" : [ ]
    }
  },
  "parent" : null
}
```

3.1.1. Response Structure

The response contains details of the application's beans. The following table describes the structure of the response:

Path	Type	Description
`contextId`	`String`	ID of the application context.
`beans.*`	`Object`	Beans in the application context keyed by name.
`beans.*.aliases`	`Array`	Names of any aliases.
`beans.*.scope`	`String`	Scope of the bean.
`beans.*.type`	`String`	Fully qualified type of the bean.
`beans.*.resource`	`String`	Resource in which the bean was defined, if any.
`beans.*.dependencies`	`Array`	Names of any dependencies.
`parent`	`Object`	Beans in the parent application context, if any.

Chapter 4. Conditions Evaluation Report (`conditions`)

The `conditions` endpoint provides information about the evaluation of conditions on configuration and auto-configuration classes.

4.1. Retrieving the Report

To retrieve the report, make a `GET` request to `/actuator/conditions`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/conditions' -i
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 2979

{
  "positiveMatches" : {
    "WebEndpointAutoConfiguration#endpointMediaTypes" : [ {
      "condition" : "OnBeanCondition",
      "message" : "@ConditionalOnMissingBean (types:
org.springframework.boot.actuate.endpoint.web.EndpointMediaTypes; SearchStrategy: all)
did not find any beans"
    } ],
    "EndpointAutoConfiguration#endpointCachingOperationInvokerAdvisor" : [ {
      "condition" : "OnBeanCondition",
      "message" : "@ConditionalOnMissingBean (types:
org.springframework.boot.actuate.endpoint.cache.CachingOperationInvokerAdvisor;
SearchStrategy: all) did not find any beans"
    } ],
    "WebEndpointAutoConfiguration" : [ {
      "condition" : "OnWebApplicationCondition",
      "message" : "@ConditionalOnWebApplication (required) found 'session' scope"
    }, {
      "condition" : "OnPropertyCondition",
      "message" : "@ConditionalOnProperty (management.endpoints.web.enabled) matched"
    } ]
  },
  "negativeMatches" : {
    "JacksonAutoConfiguration.JodaDateTimeJacksonConfiguration" : {
      "notMatched" : [ {
        "condition" : "OnClassCondition",
        "message" : "@ConditionalOnClass did not find required classes
'com.fasterxml.jackson.datatype.joda.ser.DateTimeSerializer',
```

```
'com.fasterxml.jackson.datatype.joda.cfg.JacksonJodaDateFormat'"
      } ],
      "matched" : [ ]
    },
    "GsonHttpMessageConvertersConfiguration.GsonHttpMessageConverterConfiguration" : {
      "notMatched" : [ {
        "condition" :
"GsonHttpMessageConvertersConfiguration.PreferGsonOrJacksonAndJsonbUnavailableConditio
n",
        "message" : "AnyNestedCondition 0 matched 2 did not; NestedCondition on
GsonHttpMessageConvertersConfiguration.PreferGsonOrJacksonAndJsonbUnavailableCondition
.JacksonJsonbUnavailable NoneNestedConditions 1 matched 1 did not; NestedCondition on
GsonHttpMessageConvertersConfiguration.JacksonAndJsonbUnavailable.JsonbPreferred
@ConditionalOnProperty (spring.http.converters.preferred-json-mapper=jsonb) did not
find property 'spring.http.converters.preferred-json-mapper'; NestedCondition on
GsonHttpMessageConvertersConfiguration.JacksonAndJsonbUnavailable.JacksonAvailable
@ConditionalOnBean (types:
org.springframework.http.converter.json.MappingJackson2HttpMessageConverter;
SearchStrategy: all) found bean 'mappingJackson2HttpMessageConverter'; NestedCondition
on
GsonHttpMessageConvertersConfiguration.PreferGsonOrJacksonAndJsonbUnavailableCondition
.GsonPreferred @ConditionalOnProperty (spring.http.converters.preferred-json-
mapper=gson) did not find property 'spring.http.converters.preferred-json-mapper'"
      } ],
      "matched" : [ ]
    },
    "JsonbHttpMessageConvertersConfiguration" : {
      "notMatched" : [ {
        "condition" : "OnClassCondition",
        "message" : "@ConditionalOnClass did not find required class
'javax.json.bind.Jsonb'"
      } ],
      "matched" : [ ]
    }
  },
  "unconditionalClasses" : [
"org.springframework.boot.autoconfigure.context.PropertyPlaceholderAutoConfiguration"
]
}
```


4.1.1. Response Structure

The response contains details of the application's condition evaluation. The following table describes the structure of the response:

Path	Type	Description
`positiveMatches`	`Object`	Classes and methods with conditions that were matched.
`positiveMatches.*.[].condition`	`String`	Name of the condition.

Path	Type	Description
`positiveMatches.*.[].message`	`String`	Details of why the condition was matched.
`negativeMatches`	`Object`	Classes and methods with conditions that were not matched.
`negativeMatches.*.notMatched`	`Array`	Conditions that were matched.
`negativeMatches.*.notMatched.[].condition`	`String`	Name of the condition.
`negativeMatches.*.notMatched.[].message`	`String`	Details of why the condition was not matched.
`negativeMatches.*.matched`	`Array`	Conditions that were matched.
`negativeMatches.*.matched.[].condition`	`String`	Name of the condition.
`negativeMatches.*.matched.[].message`	`String`	Details of why the condition was matched.
`unconditionalClasses`	`Array`	Names of unconditional auto-configuration classes, if any.

Chapter 5. Configuration Properties (configprops)

The configprops endpoint provides information about the application's @ConfigurationProperties beans.

5.1. Retrieving the @ConfigurationProperties Bean

To retrieve the @ConfigurationProperties beans, make a GET request to /actuator/configprops, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/configprops' -i
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 1451

{
  "contextId" : "application:-1",
  "beans" : {
    "spring.http.encoding-
org.springframework.boot.autoconfigure.http.HttpEncodingProperties" : {
      "prefix" : "spring.http.encoding",
      "properties" : {
        "charset" : "UTF-8",
        "force" : false,
        "forceRequest" : false,
        "forceResponse" : false
      }
    },
    "management.endpoints.web-
org.springframework.boot.actuate.autoconfigure.endpoint.web.WebEndpointProperties" : {
      "prefix" : "management.endpoints.web",
      "properties" : {
        "exclude" : [ ],
        "pathMapping" : { },
        "basePath" : "/actuator",
        "expose" : [ "*" ]
      }
    },
    "spring.resources-org.springframework.boot.autoconfigure.web.ResourceProperties" :
{
      "prefix" : "spring.resources",
      "properties" : {
        "addMappings" : true,
```

```
        "chain" : {
          "cache" : true,
          "htmlApplicationCache" : false,
          "gzipped" : false,
          "strategy" : {
            "fixed" : {
              "enabled" : false,
              "paths" : [ "/**" ]
            },
            "content" : {
              "enabled" : false,
              "paths" : [ "/**" ]
            }
          }
        },
        "cache" : {
          "cachecontrol" : { }
        },
        "staticLocations" : [ "classpath:/META-INF/resources/",
"classpath:/resources/", "classpath:/static/", "classpath:/public/" ]
      }
    }
  },
  "parent" : null
}
```

5.1.1. Response Structure

The response contains details of the application's `@ConfigurationProperties` beans. The following table describes the structure of the response:

Path	Type	Description
`contextId`	`String`	ID of the application context.
`beans.*`	`Object`	`@ConfigurationProperties` beans keyed by bean name.
`beans.*.prefix`	`String`	Prefix applied to the names of the bean's properties.
`beans.*.properties`	`Object`	Properties of the bean as name-value pairs.
`parent`	`Object`	`@ConfigurationProperties` beans in the parent context, if any.

Chapter 6. Environment (env)

The `env` endpoint provides information about the application's `Environment`.

6.1. Retrieving the Entire Environment

To retrieve the entire environment, make a `GET` request to `/actuator/env`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/env' -i
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 799

{
  "activeProfiles" : [ ],
  "propertySources" : [ {
    "name" : "systemProperties",
    "properties" : {
      "java.runtime.name" : {
        "value" : "OpenJDK Runtime Environment"
      },
      "java.vm.version" : {
        "value" : "25.141-b15"
      },
      "java.vm.vendor" : {
        "value" : "Oracle Corporation"
      }
    }
  }, {
    "name" : "systemEnvironment",
    "properties" : {
      "JAVA_HOME" : {
        "value" : "/docker-java-home",
        "origin" : "System Environment Property \"JAVA_HOME\""
      }
    }
  }, {
    "name" : "applicationConfig: [classpath:/application.properties]",
    "properties" : {
      "com.example.cache.max-size" : {
        "value" : "1000",
        "origin" : "class path resource [application.properties]:1:29"
      }
    }
  } ]
}
```

6.1.1. Response Structure

The response contains details of the application's `Environment`. The following table describes the structure of the response:

Path	Type	Description
`activeProfiles`	`Array`	Names of the active profiles, if any.
`propertySources`	`Array`	Property sources in order of precedence.
`propertySources.[].name`	`String`	Name of the property source.

Path	Type	Description
`propertySources.[].properties`	`Object`	Properties in the property source keyed by property name.
`propertySources.[].properties.*.value`	`String`	Value of the property.
`propertySources.[].properties.*.origin`	`String`	Origin of the property, if any.

6.2. Retrieving a Single Property

To retrieve a single property, make a `GET` request to `/actuator/env/{property.name}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/env/com.example.cache.max-size' -i
```

The preceding example retrieves information about the property named `com.example.cache.max-size`. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Disposition: inline;filename=f.txt
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 445

{
  "property" : {
    "source" : "applicationConfig: [classpath:/application.properties]",
    "value" : "1000"
  },
  "activeProfiles" : [ ],
  "propertySources" : [ {
    "name" : "systemProperties"
  }, {
    "name" : "systemEnvironment"
  }, {
    "name" : "applicationConfig: [classpath:/application.properties]",
    "property" : {
      "value" : "1000",
      "origin" : "class path resource [application.properties]:1:29"
    }
  } ]
}
```

6.2.1. Response Structure

The response contains details of the requested property. The following table describes the structure of the response:

Path	Type	Description
`property`	`Object`	Property from the environment, if found.
`property.source`	`String`	Name of the source of the property.
`property.value`	`String`	Value of the property.
`activeProfiles`	`Array`	Names of the active profiles, if any.
`propertySources`	`Array`	Property sources in order of precedence.
`propertySources.[].name`	`String`	Name of the property source.
`propertySources.[].property`	`Object`	Property in the property source, if any.
`propertySources.[].property.value`	`String`	Value of the property.
`propertySources.[].property.origin`	`String`	Origin of the property, if any.

Chapter 7. Flyway (`flyway`)

The `flyway` endpoint provides information about database migrations performed by Flyway.

7.1. Retrieving the Migrations

To retrieve the migrations, make a `GET` request to `/actuator/flyway`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/flyway' -i
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 345

{
  "flyway" : {
    "migrations" : [ {
      "type" : "SQL",
      "checksum" : 0,
      "version" : "1",
      "description" : "init",
      "script" : "V1__init.sql",
      "state" : "SUCCESS",
      "installedBy" : "SA",
      "installedOn" : "2017-11-30T04:54:44.299+0000",
      "installedRank" : 1,
      "executionTime" : 0
    } ]
  }
}
```

7.1.1. Response Structure

The response contains details of the application's Flyway migrations. The following table describes the structure of the response:

Path	Type	Description
`*.migrations`	`Array`	Migrations performed by the Flyway instance, keyed by bean name.
`*.migrations.[].checksum`	`Number`	Checksum of the migration, if any.
`*.migrations.[].description`	`String`	Description of the migration, if any.

Path	Type	Description
*.migrations.[].executionTime	Number	Execution time in milliseconds of an applied migration.
*.migrations.[].installedBy	String	User that installed the applied migration, if any.
*.migrations.[].installedOn	String	Timestamp of when the applied migration was installed, if any.
*.migrations.[].installedRank	Number	Rank of the applied migration, if any. Later migrations have higher ranks.
*.migrations.[].script	String	Name of the script used to execute the migration, if any.
*.migrations.[].state	String	State of the migration. (PENDING, ABOVE_TARGET, BELOW_BASELINE, BASELINE, IGNORED, MISSING_SUCCESS, MISSING_FAILED, SUCCESS, FAILED, OUT_OF_ORDER, FUTURE_SUCCESS, FUTURE_FAILED, OUTDATED, SUPERSEEDED)
*.migrations.[].type	String	Type of the migration. (SCHEMA, BASELINE, SQL, JDBC, SPRING_JDBC, CUSTOM)
*.migrations.[].version	String	Version of the database after applying the migration, if any.

Chapter 8. Health (`health`)

The `health` endpoint provides detailed information about the health of the application.

8.1. Retrieving the Health

To retrieve the health of the application, make a `GET` request to `/actuator/health`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/health' -i
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 385

{
  "status" : "UP",
  "details" : {
    "diskSpaceHealthIndicator" : {
      "status" : "UP",
      "details" : {
        "total" : 59258765312,
        "free" : 28445601792,
        "threshold" : 10485760
      }
    },
    "dataSourceHealthIndicator" : {
      "status" : "UP",
      "details" : {
        "database" : "HSQL Database Engine",
        "hello" : 1
      }
    }
  }
}
```

8.1.1. Response Structure

The response contains details of the health of the application. The following table describes the structure of the response:

Path	Type	Description
`status`	`String`	Overall status of the application.

Path	Type	Description
`details`	`Object`	Details of the health of the application (only included when `management.endpoint.health.show-details` is `true`).
`details.*.status`	`String`	Status of a specific part of the application.
`details.*.details`	`Object`	Details of the health of a specific part of the application.

Chapter 9. Heap Dump (heapdump)

The heapdump endpoint provides a heap dump from the application's JVM.

9.1. Retrieving the Heap Dump

To retrieve the heap dump, make a GET request to /actuator/heapdump. The response is binary data in HPROF format and can be large. Typically, you should save the response to disk for subsequent analysis. When using curl, this can be achieved by using the -O option, as shown in the following example:

```
$ curl 'http://localhost:8080/actuator/heapdump' -O
```

The preceding example results in a file named heapdump being written to the current working directory.

Chapter 10. Info (info)

The info endpoint provides general information about the application.

10.1. Retrieving the Info

To retrieve the information about the application, make a GET request to /actuator/info, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/info' -i
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 241

{
  "git" : {
    "commit" : {
      "time" : "a883-12-07T16:57:33.000+0000",
      "id" : "df027cf"
    },
    "branch" : "master"
  },
  "build" : {
    "version" : "1.0.3",
    "artifact" : "application",
    "group" : "com.example"
  }
}
```

10.1.1. Response Structure

The response contains general information about the application. Each section of the response is contributed by an InfoContributor. Spring Boot provides build and git contributions.

build Response Structure

The following table describe the structure of the build section of the response:

Path	Type	Description
artifact	String	Artifact ID of the application, if any.
group	String	Group ID of the application, if any.
name	String	Name of the application, if any.

Path	Type	Description
`version`	`String`	Version of the application, if any.
`time`	`Varies`	Timestamp of when the application was built, if any.

`git` Response Structure

The following table describes the structure of the `git` section of the response:

Path	Type	Description
`branch`	`String`	Name of the Git branch, if any.
`commit`	`Object`	Details of the Git commit, if any.
`commit.time`	`Varies`	Timestamp of the commit, if any.
`commit.id`	`String`	ID of the commit, if any.

Chapter 11. Liquibase (`liquibase`)

The `liquibase` endpoint provides information about database change sets applied by Liquibase.

11.1. Retrieving the Changes

To retrieve the changes, make a `GET` request to `/actuator/liquibase`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/liquibase' -i
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 532

{
  "liquibase" : {
    "changeSets" : [ {
      "author" : "marceloverdijk",
      "changeLog" : "classpath:/db/changelog/db.changelog-master.yaml",
      "comments" : "",
      "contexts" : [ ],
      "dateExecuted" : "2017-11-30T04:53:52.614+0000",
      "deploymentId" : "2017632603",
      "description" : "createTable tableName=customer",
      "execType" : "EXECUTED",
      "id" : "1",
      "labels" : [ ],
      "checksum" : "7:0cfbff0a94f5ba816ab56eaca6b8affc",
      "orderExecuted" : 1,
      "tag" : null
    } ]
  }
}
```

11.1.1. Response Structure

The response contains details of the application's Liquibase change sets. The following table describes the structure of the response:

Path	Type	Description
`*.changeSets`	`Array`	Change sets made by the Liquibase beans, keyed by bean name.

Path	Type	Description
`*.changeSets[].author`	`String`	Author of the change set.
`*.changeSets[].changeLog`	`String`	Change log that contains the change set.
`*.changeSets[].comments`	`String`	Comments on the change set.
`*.changeSets[].contexts`	`Array`	Contexts of the change set.
`*.changeSets[].dateExecuted`	`String`	Timestamp of when the change set was executed.
`*.changeSets[].deploymentId`	`String`	ID of the deployment that ran the change set.
`*.changeSets[].description`	`String`	Description of the change set.
`*.changeSets[].execType`	`String`	Execution type of the change set (`EXECUTED`, `FAILED`, `SKIPPED`, `RERAN`, `MARK_RAN`).
`*.changeSets[].id`	`String`	ID of the change set.
`*.changeSets[].labels`	`Array`	Labels associated with the change set.
`*.changeSets[].checksum`	`String`	Checksum of the change set.
`*.changeSets[].orderExecuted`	`Number`	Order of the execution of the change set.
`*.changeSets[].tag`	`Null`	Tag associated with the change set.

Chapter 12. Log File (`logfile`)

The `logfile` endpoint provides access to the contents of the application's log file.

12.1. Retrieving the Log File

To retrieve the log file, make a `GET` request to `/actuator/logfile`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/logfile' -i
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Content-Type: text/plain
Content-Length: 4723


  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::

2017-08-08 17:12:30.910  INFO 19866 --- [           main]
s.f.SampleWebFreeMarkerApplication       : Starting SampleWebFreeMarkerApplication on
host.local with PID 19866
2017-08-08 17:12:30.913  INFO 19866 --- [           main]
s.f.SampleWebFreeMarkerApplication       : No active profile set, falling back to
default profiles: default
2017-08-08 17:12:30.952  INFO 19866 --- [           main]
ConfigServletWebServerApplicationContext : Refreshing
org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicati
onContext@76b10754: startup date [Tue Aug 08 17:12:30 BST 2017]; root of context
hierarchy
2017-08-08 17:12:31.878  INFO 19866 --- [           main]
o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080
(http)
2017-08-08 17:12:31.889  INFO 19866 --- [           main]
o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2017-08-08 17:12:31.890  INFO 19866 --- [           main]
org.apache.catalina.core.StandardEngine  : Starting Servlet Engine: Apache
Tomcat/8.5.16
2017-08-08 17:12:31.978  INFO 19866 --- [ost-startStop-1]
o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded
WebApplicationContext
```

```
2017-08-08 17:12:31.978  INFO 19866 --- [ost-startStop-1]
o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization
completed in 1028 ms
2017-08-08 17:12:32.080  INFO 19866 --- [ost-startStop-1]
o.s.b.w.servlet.ServletRegistrationBean  : Mapping servlet: 'dispatcherServlet' to [/]
2017-08-08 17:12:32.084  INFO 19866 --- [ost-startStop-1]
o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'characterEncodingFilter'
to: [/*]
2017-08-08 17:12:32.084  INFO 19866 --- [ost-startStop-1]
o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'hiddenHttpMethodFilter'
to: [/*]
2017-08-08 17:12:32.084  INFO 19866 --- [ost-startStop-1]
o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'httpPutFormContentFilter'
to: [/*]
2017-08-08 17:12:32.084  INFO 19866 --- [ost-startStop-1]
o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'requestContextFilter' to:
[/*]
2017-08-08 17:12:32.349  INFO 19866 --- [           main]
s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice:
org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicati
onContext@76b10754: startup date [Tue Aug 08 17:12:30 BST 2017]; root of context
hierarchy
2017-08-08 17:12:32.420  INFO 19866 --- [           main]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public
org.springframework.http.ResponseEntity<java.util.Map<java.lang.String,
java.lang.Object>>
org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.error(ja
vax.servlet.http.HttpServletRequest)
2017-08-08 17:12:32.421  INFO 19866 --- [           main]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}"
onto public org.springframework.web.servlet.ModelAndView
org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.errorHtm
l(javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServletResponse)
2017-08-08 17:12:32.444  INFO 19866 --- [           main]
o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/webjars/**] onto handler
of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2017-08-08 17:12:32.444  INFO 19866 --- [           main]
o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**] onto handler of type
[class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2017-08-08 17:12:32.471  INFO 19866 --- [           main]
o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**/favicon.ico] onto
handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2017-08-08 17:12:32.600  INFO 19866 --- [           main]
o.s.w.s.v.f.FreeMarkerConfigurer         : ClassTemplateLoader for Spring macros added
to FreeMarker configuration
2017-08-08 17:12:32.681  INFO 19866 --- [           main]
o.s.j.e.a.AnnotationMBeanExporter        : Registering beans for JMX exposure on
startup
2017-08-08 17:12:32.744  INFO 19866 --- [           main]
o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http)
```

```
2017-08-08 17:12:32.750  INFO 19866 --- [           main]
s.f.SampleWebFreeMarkerApplication       : Started SampleWebFreeMarkerApplication in
2.172 seconds (JVM running for 2.479)
```

12.2. Retrieving Part of the Log File

Retrieving part of the log file is not supported when using Jersey.

To retrieve part of the log file, make a `GET` request to `/actuator/logfile` by using the `Range` header, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/logfile' -i \
    -H 'Range: bytes=0-1023'
```

The preceding example retrieves the first 1024 bytes of the log file. The resulting response is similar to the following:

```
HTTP/1.1 206 Partial Content
Accept-Ranges: bytes
Content-Type: text/plain
Content-Range: bytes 0-1023/4723
Content-Length: 1024

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::

2017-08-08 17:12:30.910  INFO 19866 --- [           main]
s.f.SampleWebFreeMarkerApplication       : Starting SampleWebFreeMarkerApplication on
host.local with PID 19866
2017-08-08 17:12:30.913  INFO 19866 --- [           main]
s.f.SampleWebFreeMarkerApplication       : No active profile set, falling back to
default profiles: default
2017-08-08 17:12:30.952  INFO 19866 --- [           main]
ConfigServletWebServerApplicationContext : Refreshing
org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicati
onContext@76b10754: startup date [Tue Aug 08 17:12:30 BST 2017]; root of context
hierarchy
2017-08-08 17:12:31.878  INFO 19866 --- [           main]
o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(
```

Chapter 13. Loggers (`loggers`)

The `loggers` endpoint provides access to the application's loggers and the configuration of their levels.

13.1. Retrieving All Loggers

To retrieve the application's loggers, make a `GET` request to `/actuator/loggers`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/loggers' -i
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 283

{
  "levels" : [ "OFF", "FATAL", "ERROR", "WARN", "INFO", "DEBUG", "TRACE" ],
  "loggers" : {
    "ROOT" : {
      "configuredLevel" : "INFO",
      "effectiveLevel" : "INFO"
    },
    "com.example" : {
      "configuredLevel" : "DEBUG",
      "effectiveLevel" : "DEBUG"
    }
  }
}
```

13.1.1. Response Structure

The response contains details of the application's loggers. The following table describes the structure of the response:

Path	Type	Description
`levels`	`Array`	Levels support by the logging system.
`loggers`	`Object`	Loggers keyed by name.
`loggers.*.configuredLevel`	`String`	Configured level of the logger, if any.
`loggers.*.effectiveLevel`	`String`	Effective level of the logger.

13.2. Retrieving a Single Logger

To retrieve a single logger, make a `GET` request to `/actuator/loggers/{logger.name}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/loggers/com.example' -i
```

The preceding example retrieves information about the logger named `com.example`. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Disposition: inline;filename=f.txt
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 61

{
  "configuredLevel" : "INFO",
  "effectiveLevel" : "INFO"
}
```

13.2.1. Response Structure

The response contains details of the requested logger. The following table describes the structure of the response:

Path	Type	Description
`configuredLevel`	`String`	Configured level of the logger, if any.
`effectiveLevel`	`String`	Effective level of the logger.

13.3. Setting a Log Level

To set the level of a logger, make a `POST` request to `/actuator/loggers/{logger.name}` with a JSON body that specifies the configured level for the logger, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/loggers/com.example' -i -X POST \
    -H 'Content-Type: application/json' \
    -d '{"configuredLevel":"debug"}'
```

The preceding example sets the `configuredLevel` of the `com.example` logger to `DEBUG`.

13.3.1. Request Structure

The request specifies the desired level of the logger. The following table describes the structure of

the request:

Path	Type	Description
`configuredLevel`	`String`	Level for the logger. May be omitted to clear the level.

13.4. Clearing a Log Level

To clear the level of a logger, make a `POST` request to `/actuator/loggers/{logger.name}` with a JSON body containing an empty object, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/loggers/com.example' -i -X POST \
    -H 'Content-Type: application/json' \
    -d '{}'
```

The preceding example clears the configured level of the `com.example` logger.

Chapter 14. Metrics (`metrics`)

The `metrics` endpoint provides access to application metrics.

14.1. Retrieving Metric Names

To retrieve the names of the available metrics, make a `GET` request to `/actuator/metrics`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/metrics' -i
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 154

{
  "names" : [ "jvm.memory.max", "jvm.memory.committed", "jvm.buffer.memory.used",
"jvm.memory.used", "jvm.buffer.count", "jvm.buffer.total.capacity" ]
}
```

14.1.1. Response Structure

The response contains details of the metric names. The following table describes the structure of the response:

Path	Type	Description
`names`	`Array`	Names of the known metrics.

14.2. Retrieving a Metric

To retrieve a metric, make a `GET` request to `/actuator/metrics/{metric.name}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/metrics/jvm.memory.max' -i
```

The preceding example retrieves information about the metric named `jvm.memory.max`. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Disposition: inline;filename=f.txt
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 390

{
  "name" : "jvm.memory.max",
  "measurements" : [ {
    "statistic" : "Value",
    "value" : 2.373451775E9
  } ],
  "availableTags" : [ {
    "tag" : "area",
    "values" : [ "heap", "heap", "heap", "nonheap", "nonheap", "nonheap" ]
  }, {
    "tag" : "id",
    "values" : [ "PS Old Gen", "PS Survivor Space", "PS Eden Space", "Code Cache",
"Compressed Class Space", "Metaspace" ]
  } ]
}
```

14.2.1. Query Parameters

The endpoint uses query parameters to drill down into a metric by using its tags. The following table shows the single supported query parameter:

Parameter	Description
`tag`	A tag to use for drill-down in the form `name:value`.

14.2.2. Response structure

The response contains details of the metric. The following table describes the structure of the response:

Path	Type	Description
`name`	`String`	Name of the metric
`measurements`	`Array`	Measurements of the metric
`measurements[].statistic`	`String`	Statistic of the measurement. (`Total`, `TotalTime`, `Count`, `Max`, `Value`, `Unknown`, `ActiveTasks`, `Duration`).
`measurements[].value`	`Number`	Value of the measurement.
`availableTags`	`Array`	Tags that are available for drill-down.

Path	Type	Description
`availableTags[].tag`	`String`	Name of the tag.
`availableTags[].values`	`Array`	Possible values of the tag.

14.3. Drilling Down

To drill down into a metric, make a `GET` request to `/actuator/metrics/{metric.name}` using the `tag` query parameter, as shown in the following curl-based example:

```
$ curl
'http://localhost:8080/actuator/metrics/jvm.memory.max?tag=area%3Anonheap&tag=id%3ACom
pressed+Class+Space' -i
```

The preceding example retrieves the `jvm.memory.max` metric, where the `area` tag has a value of `nonheap` and the `id` attribute has a value of `Compressed Class Space`. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Disposition: inline;filename=f.txt
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 141

{
  "name" : "jvm.memory.max",
  "measurements" : [ {
    "statistic" : "Value",
    "value" : 1.073741824E9
  } ],
  "availableTags" : [ ]
}
```

Chapter 15. Prometheus (`prometheus`)

The `prometheus` endpoint provides Spring Boot application's metrics in the format required for scraping by a Prometheus server.

15.1. Retrieving the Metrics

To retrieve the metrics, make a `GET` request to `/actuator/prometheus`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/prometheus' -i
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: text/plain;version=0.0.4;charset=utf-8
Content-Length: 2337

# HELP jvm_buffer_count An estimate of the number of buffers in the pool
# TYPE jvm_buffer_count gauge
jvm_buffer_count{id="direct",} 5.0
jvm_buffer_count{id="mapped",} 0.0
# HELP jvm_buffer_memory_used_bytes An estimate of the memory that the Java virtual
machine is using for this buffer pool
# TYPE jvm_buffer_memory_used_bytes gauge
jvm_buffer_memory_used_bytes{id="direct",} 33967.0
jvm_buffer_memory_used_bytes{id="mapped",} 0.0
# HELP jvm_memory_max_bytes The maximum amount of memory in bytes that can be used for
memory management
# TYPE jvm_memory_max_bytes gauge
jvm_memory_max_bytes{area="nonheap",id="Code Cache",} 2.5165824E8
jvm_memory_max_bytes{area="nonheap",id="Metaspace",} -1.0
jvm_memory_max_bytes{area="nonheap",id="Compressed Class Space",} 1.073741824E9
jvm_memory_max_bytes{area="heap",id="PS Eden Space",} 3.0670848E8
jvm_memory_max_bytes{area="heap",id="PS Survivor Space",} 2.5165824E7
jvm_memory_max_bytes{area="heap",id="PS Old Gen",} 7.16177408E8
# HELP jvm_memory_committed_bytes The amount of memory in bytes that is committed for
the Java virtual machine to use
# TYPE jvm_memory_committed_bytes gauge
jvm_memory_committed_bytes{area="nonheap",id="Code Cache",} 4.4040192E7
jvm_memory_committed_bytes{area="nonheap",id="Metaspace",} 1.35094272E8
jvm_memory_committed_bytes{area="nonheap",id="Compressed Class Space",} 1.8743296E7
jvm_memory_committed_bytes{area="heap",id="PS Eden Space",} 3.06184192E8
jvm_memory_committed_bytes{area="heap",id="PS Survivor Space",} 2.5165824E7
jvm_memory_committed_bytes{area="heap",id="PS Old Gen",} 1.30023424E8
# HELP jvm_buffer_total_capacity_bytes An estimate of the total capacity of the
buffers in this pool
# TYPE jvm_buffer_total_capacity_bytes gauge
jvm_buffer_total_capacity_bytes{id="direct",} 33966.0
jvm_buffer_total_capacity_bytes{id="mapped",} 0.0
# HELP jvm_memory_used_bytes The amount of used memory
# TYPE jvm_memory_used_bytes gauge
jvm_memory_used_bytes{area="nonheap",id="Code Cache",} 4.3791552E7
jvm_memory_used_bytes{area="nonheap",id="Metaspace",} 1.28321896E8
jvm_memory_used_bytes{area="nonheap",id="Compressed Class Space",} 1.721628E7
jvm_memory_used_bytes{area="heap",id="PS Eden Space",} 5.4957944E7
jvm_memory_used_bytes{area="heap",id="PS Survivor Space",} 2898200.0
jvm_memory_used_bytes{area="heap",id="PS Old Gen",} 1.17560352E8
```

Chapter 16. Scheduled Tasks (scheduledtasks)

The scheduledtasks endpoint provides information about the application's scheduled tasks.

16.1. Retrieving the Scheduled Tasks

To retrieve the scheduled tasks, make a GET request to /actuator/scheduledtasks, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/scheduledtasks' -i
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 451

{
  "cron" : [ {
    "runnable" : {
      "target" : "com.example.Processor.processOrders"
    },
    "expression" : "0 0 0/3 1/1 * ?"
  } ],
  "fixedDelay" : [ {
    "runnable" : {
      "target" : "com.example.Processor.purge"
    },
    "initialDelay" : 5000,
    "interval" : 5000
  } ],
  "fixedRate" : [ {
    "runnable" : {
      "target" : "com.example.Processor.retrieveIssues"
    },
    "initialDelay" : 10000,
    "interval" : 3000
  } ]
}
```

16.1.1. Response Structure

The response contains details of the application's scheduled tasks. The following table describes the structure of the response:

Path	Type	Description
`cron`	`Array`	Cron tasks, if any.
`cron.[]runnable.target`	`String`	Target that will be executed.
`cron.[].expression`	`String`	Cron expression.
`fixedDelay`	`Array`	Fixed delay tasks, if any.
`fixedDelay.[]runnable.target`	`String`	Target that will be executed.
`fixedDelay.[].initialDelay`	`Number`	Delay, in milliseconds, before first execution.
`fixedDelay.[].interval`	`Number`	Interval, in milliseconds, between the end of the last execution and the start of the next.
`fixedRate`	`Array`	Fixed rate tasks, if any.
`fixedRate.[].runnable.target`	`String`	Target that will be executed.
`fixedRate.[].interval`	`Number`	Interval, in milliseconds, between the start of each execution.
`fixedRate.[].initialDelay`	`Number`	Delay, in milliseconds, before first execution.

Chapter 17. Sessions (`sessions`)

The `sessions` endpoint provides information about the application's HTTP sessions that are managed by Spring Session.

17.1. Retrieving Sessions

To retrieve the sessions, make a `GET` request to `/actuator/sessions`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/sessions?username=alice' -i
```

The preceding examples retrieves all of the sessions for the user whose username is `alice`.

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 675

{
  "sessions" : [ {
    "id" : "0fedd4c6-7350-4699-b3ed-aba74969334b",
    "attributeNames" : [ ],
    "creationTime" : 1511974491283,
    "lastAccessedTime" : 1512017646283,
    "maxInactiveInterval" : 1800,
    "expired" : false
  }, {
    "id" : "b3d10e00-106b-471c-a034-f6b1db7c9c05",
    "attributeNames" : [ ],
    "creationTime" : 1512010491284,
    "lastAccessedTime" : 1512017679284,
    "maxInactiveInterval" : 1800,
    "expired" : false
  }, {
    "id" : "4db5efcc-99cb-4d05-a52c-b49acfbb7ea9",
    "attributeNames" : [ ],
    "creationTime" : 1511999691284,
    "lastAccessedTime" : 1512017654284,
    "maxInactiveInterval" : 1800,
    "expired" : false
  } ]
}
```

17.1.1. Query Parameters

The endpoint uses query parameters to limit the sessions that it returns. The following table shows the single required query parameter:

Parameter	Description
`username`	Name of the user.

17.1.2. Response Structure

The response contains details of the matching sessions. The following table describes the structure of the response:

Path	Type	Description
`sessions`	`Array`	Sessions for the given username.
`sessions.[].id`	`String`	ID of the session.
`sessions.[].attributeNames`	`Array`	Names of the attributes stored in the session.
`sessions.[].creationTime`	`Number`	Timestamp of when the session was created.
`sessions.[].lastAccessedTime`	`Number`	Timestamp of when the session was last accessed.
`sessions.[].maxInactiveInterval`	`Number`	Maximum permitted period of inactivity, in seconds, before the session will expire.
`sessions.[].expired`	`Boolean`	Whether the session has expired.

17.2. Retrieving a Single Session

To retrieve a single session, make a `GET` request to `/actuator/sessions/{id}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/sessions/4db5efcc-99cb-4d05-a52c-b49acfbb7ea9'
-i
```

The preceding example retrieves the session with the `id` of `4db5efcc-99cb-4d05-a52c-b49acfbb7ea9`. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 202

{
  "id" : "4db5efcc-99cb-4d05-a52c-b49acfbb7ea9",
  "attributeNames" : [ ],
  "creationTime" : 1511999691284,
  "lastAccessedTime" : 1512017654284,
  "maxInactiveInterval" : 1800,
  "expired" : false
}
```

17.2.1. Response Structure

The response contains details of the requested session. The following table describes the structure of the response:

Path	Type	Description
`id`	`String`	ID of the session.
`attributeNames`	`Array`	Names of the attributes stored in the session.
`creationTime`	`Number`	Timestamp of when the session was created.
`lastAccessedTime`	`Number`	Timestamp of when the session was last accessed.
`maxInactiveInterval`	`Number`	Maximum permitted period of inactivity, in seconds, before the session will expire.
`expired`	`Boolean`	Whether the session has expired.

17.3. Deleting a Session

To delete a session, make a `DELETE` request to `/actuator/sessions/{id}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/sessions/4db5efcc-99cb-4d05-a52c-b49acfbb7ea9'
-i -X DELETE
```

The preceding example deletes the session with the `id` of `4db5efcc-99cb-4d05-a52c-b49acfbb7ea9`.

Chapter 18. Shutdown (shutdown)

The shutdown endpoint is used to shut down the application.

18.1. Shutting Down the Application

To shut down the application, make a POST request to /actuator/shutdown, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/shutdown' -i -X POST
```

A response similar to the following is produced:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 41

{
  "message" : "Shutting down, bye..."
}
```

18.1.1. Response Structure

The response contains details of the result of the shutdown request. The following table describes the structure of the response:

Path	Type	Description
message	String	Message describing the result of the request.

Chapter 19. Thread Dump (`threaddump`)

The `threaddump` endpoint provides a thread dump from the application's JVM.

19.1. Retrieving the Thread Dump

To retrieve the thread dump, make a `GET` request to `/actuator/threaddump`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/threaddump' -i
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 6247

{
  "threads" : [ {
    "threadName" : "Thread-66",
    "threadId" : 356,
    "blockedTime" : -1,
    "blockedCount" : 0,
    "waitedTime" : -1,
    "waitedCount" : 1,
    "lockName" : null,
    "lockOwnerId" : -1,
    "lockOwnerName" : null,
    "inNative" : false,
    "suspended" : false,
    "threadState" : "TIMED_WAITING",
    "stackTrace" : [ {
      "methodName" : "sleep",
      "fileName" : "Thread.java",
      "lineNumber" : -2,
      "className" : "java.lang.Thread",
      "nativeMethod" : true
    }, {
      "methodName" : "performShutdown",
      "fileName" : "ShutdownEndpoint.java",
      "lineNumber" : 67,
      "className" : "org.springframework.boot.actuate.context.ShutdownEndpoint",
      "nativeMethod" : false
    }, {
      "methodName" : "run",
      "fileName" : null,
      "lineNumber" : -1,
      "className" :
```

```
"org.springframework.boot.actuate.context.ShutdownEndpoint$$Lambda$998/1298182295",
      "nativeMethod" : false
    }, {
      "methodName" : "run",
      "fileName" : "Thread.java",
      "lineNumber" : 748,
      "className" : "java.lang.Thread",
      "nativeMethod" : false
    } ],
    "lockedMonitors" : [ ],
    "lockedSynchronizers" : [ ],
    "lockInfo" : null
  }, {
    "threadName" : "pool-15-thread-1",
    "threadId" : 351,
    "blockedTime" : -1,
    "blockedCount" : 0,
    "waitedTime" : -1,
    "waitedCount" : 3,
    "lockName" :
"java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject@5dcc1a36",
    "lockOwnerId" : -1,
    "lockOwnerName" : null,
    "inNative" : false,
    "suspended" : false,
    "threadState" : "TIMED_WAITING",
    "stackTrace" : [ {
      "methodName" : "park",
      "fileName" : "Unsafe.java",
      "lineNumber" : -2,
      "className" : "sun.misc.Unsafe",
      "nativeMethod" : true
    }, {
      "methodName" : "parkNanos",
      "fileName" : "LockSupport.java",
      "lineNumber" : 215,
      "className" : "java.util.concurrent.locks.LockSupport",
      "nativeMethod" : false
    }, {
      "methodName" : "awaitNanos",
      "fileName" : "AbstractQueuedSynchronizer.java",
      "lineNumber" : 2078,
      "className" :
"java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject",
      "nativeMethod" : false
    }, {
      "methodName" : "take",
      "fileName" : "ScheduledThreadPoolExecutor.java",
      "lineNumber" : 1093,
      "className" :
"java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue",
```

```
      "nativeMethod" : false
    }, {
      "methodName" : "take",
      "fileName" : "ScheduledThreadPoolExecutor.java",
      "lineNumber" : 809,
      "className" :
"java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue",
      "nativeMethod" : false
    }, {
      "methodName" : "getTask",
      "fileName" : "ThreadPoolExecutor.java",
      "lineNumber" : 1074,
      "className" : "java.util.concurrent.ThreadPoolExecutor",
      "nativeMethod" : false
    }, {
      "methodName" : "runWorker",
      "fileName" : "ThreadPoolExecutor.java",
      "lineNumber" : 1134,
      "className" : "java.util.concurrent.ThreadPoolExecutor",
      "nativeMethod" : false
    }, {
      "methodName" : "run",
      "fileName" : "ThreadPoolExecutor.java",
      "lineNumber" : 624,
      "className" : "java.util.concurrent.ThreadPoolExecutor$Worker",
      "nativeMethod" : false
    }, {
      "methodName" : "run",
      "fileName" : "Thread.java",
      "lineNumber" : 748,
      "className" : "java.lang.Thread",
      "nativeMethod" : false
    } ],
    "lockedMonitors" : [ ],
    "lockedSynchronizers" : [ ],
    "lockInfo" : {
      "className" :
"java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject",
      "identityHashCode" : 1573657142
    }
  }, {
    "threadName" : "testdb housekeeper",
    "threadId" : 342,
    "blockedTime" : -1,
    "blockedCount" : 0,
    "waitedTime" : -1,
    "waitedCount" : 2,
    "lockName" :
"java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject@1c97da26",
    "lockOwnerId" : -1,
    "lockOwnerName" : null,
```

```
    "inNative" : false,
    "suspended" : false,
    "threadState" : "TIMED_WAITING",
    "stackTrace" : [ {
      "methodName" : "park",
      "fileName" : "Unsafe.java",
      "lineNumber" : -2,
      "className" : "sun.misc.Unsafe",
      "nativeMethod" : true
    }, {
      "methodName" : "parkNanos",
      "fileName" : "LockSupport.java",
      "lineNumber" : 215,
      "className" : "java.util.concurrent.locks.LockSupport",
      "nativeMethod" : false
    }, {
      "methodName" : "awaitNanos",
      "fileName" : "AbstractQueuedSynchronizer.java",
      "lineNumber" : 2078,
      "className" :
"java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject",
      "nativeMethod" : false
    }, {
      "methodName" : "take",
      "fileName" : "ScheduledThreadPoolExecutor.java",
      "lineNumber" : 1093,
      "className" :
"java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue",
      "nativeMethod" : false
    }, {
      "methodName" : "take",
      "fileName" : "ScheduledThreadPoolExecutor.java",
      "lineNumber" : 809,
      "className" :
"java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue",
      "nativeMethod" : false
    }, {
      "methodName" : "getTask",
      "fileName" : "ThreadPoolExecutor.java",
      "lineNumber" : 1074,
      "className" : "java.util.concurrent.ThreadPoolExecutor",
      "nativeMethod" : false
    }, {
      "methodName" : "runWorker",
      "fileName" : "ThreadPoolExecutor.java",
      "lineNumber" : 1134,
      "className" : "java.util.concurrent.ThreadPoolExecutor",
      "nativeMethod" : false
    }, {
      "methodName" : "run",
      "fileName" : "ThreadPoolExecutor.java",
```

```
      "lineNumber" : 624,
      "className" : "java.util.concurrent.ThreadPoolExecutor$Worker",
      "nativeMethod" : false
    }, {
      "methodName" : "run",
      "fileName" : "Thread.java",
      "lineNumber" : 748,
      "className" : "java.lang.Thread",
      "nativeMethod" : false
    } ],
    "lockedMonitors" : [ ],
    "lockedSynchronizers" : [ ],
    "lockInfo" : {
      "className" :
"java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject",
      "identityHashCode" : 479713830
    }
  } ]
}
```

19.1.1. Response Structure

The response contains details of the JVM's threads. The following table describes the structure of the response:

Path	Type	Description
`threads`	`Array`	JVM's threads.
`threads.[].blockedCount`	`Number`	Total number of times that the thread has been blocked.
`threads.[].blockedTime`	`Number`	Time in milliseconds that the thread has spent blocked. -1 if thread contention monitoring is disabled.
`threads.[].daemon`	`Boolean`	Whether the thread is a daemon thread. Only available on Java 9 or later.
`threads.[].inNative`	`Boolean`	Whether the thread is executing native code.
`threads.[].lockName`	`String`	Description of the object on which the thread is blocked, if any.
`threads.[].lockInfo`	`Object`	Object for which the thread is blocked waiting.
`threads.[].lockInfo.className`	`String`	Fully qualified class name of the lock object.

Path	Type	Description
threads.[].lockInfo.identityHashCode	Number	Identity hash code of the lock object.
threads.[].lockedMonitors	Array	Monitors locked by this thread, if any
threads.[].lockedMonitors.[].className	String	Class name of the lock object.
threads.[].lockedMonitors.[].identityHashCode	Number	Identity hash code of the lock object.
threads.[].lockedMonitors.[].lockedStackDepth	Number	Stack depth where the monitor was locked.
threads.[].lockedMonitors.[].lockedStackFrame	Object	Stack frame that locked the monitor.
threads.[].lockedSynchronizers	Array	Synchronizers locked by this thread.
threads.[].lockedSynchronizers.[].className	String	Class name of the locked synchronizer.
threads.[].lockedSynchronizers.[].identifyHashCode	Number	Identity hash code of the locked synchronizer.
threads.[].lockOwnerId	Number	ID of the thread that owns the object on which the thread is blocked. -1 if the thread is not blocked.
threads.[].lockOwnerName	Null	Name of the thread that owns the object on which the thread is blocked.
threads.[].priority	Number	Priority of the thread. Only available on Java 9 or later.
threads.[].stackTrace	Array	Stack trace of the thread.
threads.[].stackTrace.[].classLoaderName	String	Name of the class loader of the class that contains the execution point identified by this entry, if any. Only available on Java 9 or later.
threads.[].stackTrace.[].className	String	Name of the class that contains the execution point identified by this entry.
threads.[].stackTrace.[].fileName	String	Name of the source file that contains the execution point identified by this entry, if any.

Path	Type	Description
`threads.[].stackTrace.[].lineNumber`	`Number`	Line number of the execution point identified by this entry. Negative if unknown.
`threads.[].stackTrace.[].methodName`	`String`	Name of the method.
`threads.[].stackTrace.[].moduleName`	`String`	Name of the module that contains the execution point identified by this entry, if any. Only available on Java 9 or later.
`threads.[].stackTrace.[].moduleVersion`	`String`	Version of the module that contains the execution point identified by this entry, if any. Only available on Java 9 or later.
`threads.[].stackTrace.[].nativeMethod`	`Boolean`	Whether the execution point is a native method.
`threads.[].suspended`	`Boolean`	Whether the thread is suspended.
`threads.[].threadId`	`Number`	ID of the thread.
`threads.[].threadName`	`String`	Name of the thread.
`threads.[].threadState`	`String`	State of the thread (`NEW`, `RUNNABLE`, `BLOCKED`, `WAITING`, `TIMED_WAITING`, `TERMINATED`).
`threads.[].waitedCount`	`Number`	Total number of times that the thread has waited for notification.
`threads.[].waitedTime`	`Number`	Time in milliseconds that the thread has spent waiting. -1 if thread contention monitoring is disabled