Set Logging Level in Spring Boot via Environment Variable

Asked 4 years, 2 months ago Active 21 days ago Viewed 47k times

50

14

Should it be possible to set logging levels through environment variables alone in a Spring Boot application?

I don't want to use `application.properties` as I'm running on Cloud Foundry and want to pick up changes without a deploy (but after the app has restarted, or restaged to be more precise).

I've tried setting env vars like `LOGGING_LEVEL_ORG_SPRINGFRAMEWORK=TRACE` but that has no effect. Putting `logging.level.org.springframework: TRACE` in `application.properties` does work though.

java spring spring-boot

asked Dec 9 '15 at 14:19
EngineerBetter_DJ
10.6k 13 69 148

The two methods presented in the question should be equivalent based on *Relaxed Binding*. Maybe there was a bug in Spring Boot at the time (version not specified). – nobar Oct 21 '19 at 18:27

Maybe the OP changed the example value to simplify the problem description and thereby falsified the problem unintentionally. Please be aware that setting log levels via env variables only works for packages but not for classes! See my answer below for details. – Peter Wippermann Feb 20 at 20:32

10 Answers

¿No encuentras la respuesta? Pregunta en Stack Overflow en español. ×

69

+50

This is just an idea, but did you try setting

`_JAVA_OPTIONS=-Dlogging.level.org.springframework=TRACE` ?

Theoretically, this way `-Dlogging.level.org.springframework=TRACE` will be passed as default JVM argument and should affect every JVM instance in your environment.

answered Dec 12 '15 at 8:05
Timekiller
2,278 1 12 13

2 Can confirm in practice: I checked out the sample-logback project (using version 1.3.0.RELEASE) and ran `mvn spring-boot:run -Dlogging.level.org.springframework=TRACE` which produced a lot of log output. Using the all caps `-DLOGGING_LEVEL_ORG_SPRINGFRAMEWORK=TRACE` or variants thereof did nothing though. – vanOekel Dec 12 '15 at 15:11

Join Stack Overflow to learn, share knowledge, and build your career. Sign up ×

necessary. See stackoverflow.com/questions/17781405 for some more info. – Timekiller Dec 12 '15 at 16:34

I'll give that a try - this will be running via the Cloud Foundry Java Buildpack, but hopefully it will respect and pass on the values in `_JAVA_OPTIONS` – EngineerBetter_DJ Dec 13 '15 at 13:44

I'm not familiar with Java Buildpack, but they have a doc on how to use another environment variable, `JAVA_OPTS` : github.com/cloudfoundry/java-buildpack/blob/master/docs/… – Timekiller Dec 13 '15 at 14:34

I would suggest JAVA_TOOL_OPTIONS over _JAVA_OPTIONS as described here: bugs.openjdk.java.net/browse/JDK-4971166 – Michael R Apr 6 '18 at 22:22

17

I also tried to set logging level via environment variable but as already mentioned it is not possible by using environment variable with upper case name, eg. `LOGGING_LEVEL_ORG_SPRINGFRAMEWORK=DEBUG` . I also didn't want to do it via `application.properties` or `_JAVA_OPTIONS` .

After digging into class `org.springframework.boot.logging.LoggingApplicationListener` I've checked that spring boot tries to set logging level `DEBUG` to `ORG_SPRINGFRAMEWORK` package which is not real package name. So conclusion is that you can use environment variable to set logging level but it needs to be in the form:
`LOGGING_LEVEL_org.springframework=DEBUG`
or
`logging.level.org.springframework=DEBUG`

Tested on spring boot 1.5.3

answered Aug 28 '17 at 7:46
pepuch
5,350 5 44 74

I tried this with Spring Boot 1.3.8 on AWS, but it did not work unfortunately. – Wim Deblauwe Apr 5 '18 at 19:18

7

Yes, you can control logging level using environment variable. Here is how I have implemented for my Spring Boot application, deployed on Cloud Foundry platform.

In you log configuration file provide placeholder for logging level to read value from environment variable. Default is INFO.

```
<logger name="com.mycompany.apps.cf" level="${APP_LOGGING_LEVEL:-INFO}">
  <appender-ref ref="CONSOLE"/>
</logger>
```

And then, in CF deployment manifest file provide environment variable.

```
applications:
- name: my-app-name
  memory: 2048
  env:
    APP_LOGGING_LEVEL: DEBUG
```

Join Stack Overflow to learn, share knowledge, and build your career. Sign up

answered Dec 16 '15 at 16:35
Sparkle8
125 8

If you are running the app locally, use -DAPP_LOGGING_LEVEL=DEBUG – Sparkle8 Dec 16 '15 at 16:38

I did do something similar as a workaround, but it means having to know which loggers you want to be able to control at design time, which is rather limiting. – EngineerBetter_DJ Dec 17 '15 at 16:50

4

Setting log levels via environment variables can only be done for Packages but not for Classes

I had the same problem as the OP. And I was wondering why some of the users here reported that the proposed solutions worked well while others responded they didn't.
I'm on **Spring Boot 2.1** and the problem obviously changed a bit over the last years, but the current situation is as follows:

TL;DR

Setting the log level for a **package works**:

```
LOGGING_LEVEL_COM_ACME_PACKAGE=DEBUG
```

While setting the log level for a specific **class has no effect**:

```
LOGGING_LEVEL_COM_ACME_PACKAGE_CLASS=DEBUG
```

How can that be?

Have a look at Spring Boot's LoggingApplicationListener.
If you'd debug it and set a breakpoint in the highlighted code block, you'd see that the log level definition for a class `com.acme.mypackage.MyClass` becomes `com.acme.mypackage.myclass` .
So **a log level definition for a class looks exactly like a log level definition for a package.**

This is related to Spring's Relaxed Binding, which proposes an upper case notation for environment variables. Thus the typical camel case notation of a class is not available for the LoggingApplicationListener. On the other hand, I think one can't check whether the given fully-qualified path matches a class. The classloader won't find anything as long as the notation doesn't match exactly.

Thus log definitions in environment variables don't work for classes but only for packages.

edited Feb 7 at 7:49

answered Aug 23 '19 at 10:03
Peter Wippermann
2,810 2 26 42

1 This was killing me on Spring Boot 2.1.2, nothing in the documentation mentions this limitation. Currently our hack is to redeploy special .jars with debug logging enabled in `application.properties` if we want to debug individual classes. Thanks for the detailed analysis – xref Feb 18 at 19:59

Join Stack Overflow to learn, share knowledge, and build your career. Sign up

3

Starting with Spring Boot 2.0.x this works again. Tested with Spring Boot v2.0.9.RELEASE. E.g. enable connection pool debug log:

```
LOGGING_LEVEL_COM_ZAXXER=DEBUG java -jar myApp.jar
```

or Spring framework debug log:

```
LOGGING_LEVEL_ORG_SPRINGFRAMEWORK=DEBUG java -jar myApp.jar
```

or both:

```
LOGGING_LEVEL_ORG_SPRINGFRAMEWORK=DEBUG LOGGING_LEVEL_COM_ZAXXER=DEBUG java -jar
myApp.jar
```

See "Application Poperties" in Spring Boot Reference Documentation for more application properties.

edited Oct 22 '19 at 12:42

answered Jul 2 '19 at 15:49
t0r0X
2,363 20 22

1 Updated link: docs.spring.io/spring-boot/docs/current/reference/html/… – nobar Oct 21 '19 at 21:08

@nobar Forgot to say Thanks for link update :-) – t0r0X Feb 19 at 13:02

2

I would anyway suggest you to use Spring profiles:

1. Create 2 properties files:

   `application-local.properties` and `application-remote.properties`

   (profile names can be different obviously)

2. Set the logging level in each file accordingly (`logging.level.org.springframework`)
3. Run your application with `-Dspring.profiles.active=local` locally and `-Dspring.profiles.active=remote` for CF.

edited Dec 15 '15 at 21:59

answered Dec 15 '15 at 21:52
Michał Urbaniak
824 7 23

Thanks. Sadly this requires knowing the things I want to increase the verbosity of at design time, rather than run time. – EngineerBetter_DJ Dec 17 '15 at 16:49

2

Also using Spring Boot (v1.2.3) in Cloud Foundry, I've found that it is possible to adjust the root logging level using an environment variable as follows:

Join Stack Overflow to learn, share knowledge, and build your career. Sign up

Unfortunately, it does not appear to be possible to dial-down the logging level for specific packages (at least with the version of Java Buildpack and Spring Boot I am using). For example adding the following in addition to the above does not reduce the log level for Spring framework:

```
$ cf set-env <app name> LOGGING_LEVEL_ORG_SPRINGFRAMEWORK INFO
```

If you are using something like Splunk to gather your logs, you may be able to filter out the noise, however.

Another alternative which looks promising could be based on customisation of the build pack's arguments option (see here):

```
$ cf set-env <app name> '{arguments: "-logging.level.root=DEBUG -
logging.level.org.springframework=INFO"}'
```

Sadly, I couldn't get this to actually work. I certainly agree that being able to reconfigure logging levels at package level without changing the application code would be handy to get working.

answered Jan 28 '16 at 10:50
Allan Lang
100 2 7

1

In spring-boot 2.0.0, adding `--trace` works. For instance `java -jar myapp.jar --debug` or `java -jar myapp.jar --trace`

https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-logging.html#boot-features-logging-console-output

answered Jun 20 '19 at 21:00
PetroCliff
884 13 20

0

Here's an example using Logback with Janino to conditionally include different logging configs via properties or environmental variables... The base config, logback.xml is using conditionals for development console logging or production file logging... just drop the following files in `/resources/`

> logback.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration scan="true">
    <if condition='property("spring.profiles.active").contains("dev")'>
        <then>
            <include resource="org/springframework/boot/logging/logback/base.xml"/>
            <include resource="dev.xml" optional="true"/>
        </then>
    </if>
```

Join Stack Overflow to learn, share knowledge, and build your career. Sign up

```
        </then>
    </if>
</configuration>
```

dev.xml

```
<included>
    <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <charset>utf-8</charset>
            <Pattern>%-30([%p] [%c:%L]) » %m%n%rEx</Pattern>
        </encoder>
    </appender>

    <!-- CHATTY LOGGERS HERE.-->
    <logger name="org.springframework" level="DEBUG"/>

    <contextListener class="ch.qos.logback.classic.jul.LevelChangePropagator">
        <resetJUL>true</resetJUL>
    </contextListener>

    <root level="${logback.loglevel}">
        <appender-ref ref="CONSOLE"/>
    </root>

</included>
```

pro.xml

```
<included>
    <conversionRule conversionWord="wex"

converterClass="org.springframework.boot.logging.logback.WhitespaceThrowableProxyConverter

    <property name="FILE_LOG_PATTERN"
              value="%d{yyyy-MM-dd HH:mm:ss.SSS} %5p ${PID:- } --- [%t]
%-40.40logger{39} : %m%n%wex"/>
    <property name="FILE_NAME_PATTERN" value="./logs/%d{yyyy-MM-dd}-exec.log"/>

    <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>FILE_NAME_PATTERN</fileNamePattern>
            <maxHistory>7</maxHistory>
        </rollingPolicy>
        <encoder>
            <pattern>${FILE_LOG_PATTERN}</pattern>
        </encoder>
    </appender>

    <appender name="ASYNC" class="ch.qos.logback.classic.AsyncAppender">
        <queueSize>512</queueSize>
        <appender-ref ref="FILE"/>
    </appender>

    <!-- APP SPECIFIC LOGGERS HERE.-->
    <logger name="org.springframework.boot.SpringApplication" level="INFO"/>

    <root level="INFO">
```

Join Stack Overflow to learn, share knowledge, and build your career. Sign up ✕

answered Dec 17 '15 at 13:06
Eddie B
4,395 1 35 37

0

Folks can anyone explain why this is not working?

$ export LOGGING_LEVEL_COM_ACME=ERROR

For all other configuration using environment variables as an override seems to work with no issues, for example:

$ export EUREKA_CLIENT_ENABLED=false

Thanks.

answered Jan 8 '16 at 11:35
Damian O' Neill
82 3

5 Because Spring Boot doesn't use the environment variables to determine logging levels. It can use JVM system properties (specified by -Dlogging.level.blah=foo), but it doesn't seem to merge in environment variables as potential property values until after it has processed logging config. – EngineerBetter_DJ Jan 8 '16 at 15:20

In fact, it DOES work! Maybe you've had a typo, or a hidden space character, etc. E.g. `LOGGING_LEVEL_COM_ZAXXER=DEBUG java -jar myApp.jar` increases the log level for classes belonging to the Hikari Connection Pool. Tested with Spring Boot v2.0.9.RELEASE. – t0r0X Jul 2 '19 at 15:03

Join Stack Overflow to learn, share knowledge, and build your career. Sign up