Stephen Colebourne's blog

Thoughts and Musings on the world of Java and beyond

Saturday, 8 February 2014

Turning off doclint in JDK 8 Javadoc

JDK 8 includes many updates, but one is I suspect going to cause quite a few complaints - doclint for Javadoc.

Javadoc doclint

Documentation is not something most developers like writing. With Java, we were fortunate to have the Javadoc toolset built in and easy to access from day one. As such, writing Javadoc is a standard part of most developers life.

The Javadoc comments in source code use a mixture of tags, starting with @, and HTML to allow the developer to express their comment and format it nicely.

Up to JDK 7, the Javadoc tool was pretty lenient. As a developer, you could write anything that vaguely resembled HTML and the tool would rarely complain beyond warnings. Thus you could have @link references that were inaccurate (such as due to refactoring) and the tool would simply provide a warning.

With JDK 8, a new part has been added to Javadoc called *doclint* and it changes that friendly behaviour. In particular, the tool aim to get conforming W3C HTML 4.01 HTML (despite the fact that humans are very bad at matching conformance wrt HTML).

With JDK 8, you are unable to get Javadoc unless your tool meets the standards of doclint. Some of its rules are:

- no self-closed HTML tags, such as
 or <a id="x" />
- no unclosed HTML tags, such as <ul> without matching
- no invalid HTML end tags, such as </br>
- no invalid HTML attributes, based on doclint's interpretation of W3C HTML 4.01
- no duplicate HTML id attribute
- no empty HTML href attribute
- no incorrectly nested headers, such as class documentation must have <h3>, not <h4>
- no invalid HTML tags, such as List<String> (where you forgot to escape using <)
- no broken @link references
- no broken @param references, they must match the actual parameter name
- no broken @throws references, the first word must be a class name

Note that these are errors, not warnings. **Break the rules and you get no Javadoc output.**

In my opinion, this is way too strict to be the default. I have no problem with such a tool existing in Javadoc, but given the history of Javadoc, errors like this should be opt-in, not opt-out. Its far better to get slightly broken Javadoc than no Javadoc.

I also haven't been able to find a list of the rules, which makes life hard. At least we can see the source code to reverse engineer them.

Turning off doclint

The magic incantation you need is `-Xdoclint:none`. This goes on the command line invoking Javadoc.

If you are running from maven, you need to use the `additionalparam` setting, as per the manual. Either add it as a global property:

```
<properties>
  <additionalparam>-Xdoclint:none</additionalparam>
</properties>
```

or add it to the maven-javadoc-plugin:

```
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-javadoc-plugin</artifactId>
    <configuration>
      <additionalparam>-Xdoclint:none</additionalparam>
    </configuration>
  </plugin>
</plugins>
```

Ant also uses `additionalparam` to pass in `-Xdoclint:none`, see the manual.

Gradle does not expose `additionalparam` but Tim Yates and Cedric Champeau advise of this solution:

```
if (JavaVersion.current().isJava8Compatible()) {
  allprojects {
    tasks.withType(Javadoc) {
      options.addStringOption('Xdoclint:none', '-quiet')
    }
  }
}
```

See also the Gradle manual.

Summary

I don't mind doclint existing, but there is no way that it should be turned on to error mode by default. Getting some Javadoc produced without hassle is far more important than pandering to the doclint style checks. In addition, it is very heavy handed with what it defines to be errors, rejecting plenty of HTML that works perfectly fine in a browser.

About Me

Stephen Colebourne

London, United Kingdom

Java developer, blogger and conference speaker

View my complete profile

Subscribe To

Posts

Comments

Blog Archive

- ► 2016 (3)
- ► 2015 (3)
- ▼ 2014 (16)
  - ► December (1)
  - ► November (7)
  - ► September (1)
  - ► August (1)
  - ► July (1)
  - ► June (1)
  - ► March (1)
  - ▼ February (3)
    - New project: ThreeTen-Extra for JDK
    - Turning off doclint in JDK 8 Javadoc
    - Exiting the JVM
- ► 2013 (2)
- ► 2012 (8)
- ► 2011 (24)
- ► 2010 (32)
- ► 2009 (19)
- ► 2008 (22)
- ► 2007 (55)
- ► 2006 (14)
- ► 2005 (14)
- ► 2004 (5)

Links

- ThreeTen / JSR-310
- Joda-Time
- Joda-Money
- Joda-Beans
- Apache Commons
- Google+
- Twitter
- LinkedIn profile
- Ohloh profile
- Privacy

Search This Blog

Sea

Labels

- bestpractice (8)
- bgga (2)
- bijava (4)
- ceylon (1)
- closures (35)
- commons (3)
- database (1)
- devoxx (10)
- elsql (1)
- fantom (5)
- fcm (12)
- firefox (1)
- fun (1)
- generics (1)
- gosu (1)
- help (1)
- java (7)
- java7 (8)
- java8 (10)
- java9 (3)
- javaedge (1)
- javaideas (47)
- javaone (7)
- jcp (26)

I've asked the maven team to disable doclint by default, and I'd suggest the same to Ant and Gradle. Unfortunately, the Oracle team seem convinced that they've made the right choice with errors by default and their use of strict HTML.

Comments welcome, but please note that non-specific "it didn't work for me" comments should be at Stack Overflow or Java Ranch, not here!

Posted by Stephen Colebourne at 23:42 G+1 +22 Recommend this on Google

Tags: java8

35 comments:

James Keesey 9 February 2014 at 03:04

How can they possibly consider this an acceptible change? Ths amount of currently valid documentation that will fail now will be immense. I know that much of my own documentation will be invalid as I write XHTML (I.e. I always close all tags). Some editors will not format Javadoc properly if the tags aren't closed.

So now, many, possibly most, developers will turn doclint off and never turn it back on again negating its benefits.

Reply

Replies

Darío Tórtola 3 October 2016 at 13:30

In my team the contrary is true. Most our documentation fails typically because we don't explain what we return or what the parameters are because we think them self-explanatory, or sometimes because we change the signature but forget to change the doc, etc; now, with a strict javadoc overseeing what we call "documentation", we are going to have to be more professional at that

Reply

Anonymous 9 February 2014 at 06:37

Surely they should be enforcing a subset of HTML5, not HTML4.01?

As all modern browsers support a fair chunk of HTML5.

Reply

Geoffrey De Smet 9 February 2014 at 11:53

I think it's a good thing they decided to fail-fast on invalid javadocs. But since when are "self-closed HTML tags, such as
" invalid?

Reply

Replies

Anonymous 9 February 2014 at 12:59

 is invalid in HTML4, and has always been. HTML is not XML. Browsers are very lenient, of course.

This stems from a complicated relationship between HTML and XHTML. An old but relevant article on the subject: http://hixie.ch/advocacy/xhtml

HTML5 fixes this issue (both
 and
 are fine, among thousands of other such things), so it's a mystery why Javadoc insists on HTML4.

Unknown 30 June 2014 at 04:42

The excuse they give in the docs is that they use frameset, but that excuse is invalid for two reasons: (1) surely even frameset works in HTML5; (2) even if it didn't, surely that one page with the frameset could be HTML4 while the rest could be HTML5.

Reply

Geoffrey De Smet 10 February 2014 at 09:42

Agreed: they need to support a subset of HTML 5, not HTML 4, especially since they fail-fast by default and HTML 4 doesn't support

.

But the decision to fail-fast by default on invalid input is a good decision I think.

Reply

Replies

Unknown 16 August 2016 at 04:56

Isn't HTML4 a subset of HTML5?

Reply

Werner Keil 19 March 2014 at 00:09

- jep (1)
- joda (18)
- jodabeans (2)
- jodaconvert (2)
- jodamoney (3)
- jodaprimitives (1)
- jodatime (17)
- jsr275 (1)
- jsr310 (13)
- jvm (2)
- jvmlang (5)
- kijaro (1)
- kotlin (2)
- nojava7 (28)
- opengamma (1)
- oracle (1)
- patents (1)
- properties (9)
- scala (4)
- serialization (1)
- splitjcp (1)
- thedeal (1)
- threeten (2)
- timezone (5)

Looks like it won't work with Maven 3.0.4, what is the minimum Maven and Java 8 build to actually accept -Xdoclint:none?

Besides styling and font of Java 8 JavaDoc look extremely bad in every browser, even the Spec Leads of some Platform JSRs avoid it for that reason...

Reply

Replies

Werner Keil 25 March 2014 at 13:35

I applied both in a POM and it ended up working with a Maven 3.x version and Java 8 close to Final. JavaDoc is still bad and someone from the Frontend team at Oracle shared the issue. The problem is, that almost every major browser shows many characters like "a" as an ugly, unreadable chunk (the upper part of the letter running into the lower one like it's melting ;-/) So far Firefox seems to handle it best, but all other browsers I checked (Chrome, IE, Opera) look horrible.

Reply

Anonymous 20 March 2014 at 10:36

I've seen on StackOverflow (link) that with JDK8 you must now add either a caption or a summary if you use HTML tables inside Javadoc comments. Why would this be required? Isn't a <table> without a caption or a summary perfectly legal HTML?

Reply

Replies

Stephen Colebourne 20 March 2014 at 20:29

Looks like its an accessibility feature. http://www.w3schools.com/tags/att_table_summary.asp . Big corporates like Oracle tend to worry about such things.

Reply

Guy McArthur 27 March 2014 at 23:40

There is an XML flavor of HTML 4 (but not HTML5) called XHTML 1.0, so something like
might well be what you intended to output....

Reply

Nathan Green 9 April 2014 at 03:24

I wish they'd ditch HTML in favor of markdown or wiki syntax or something. I have to be able to read the docs in source before they get stuffed into a full-blown HTML page, and raw HTML is not human-friendly at all. I see too much un- and under-commented code as it is: all this does is create a disincentive to write Javadoc at all. It would make more sense to fail when public classes lack Javadocs entirely than to do what they've done.

Reply

Replies

Anonymous 28 July 2015 at 02:48

+1. Never gonna happen though.

Wim Deblauwe 13 November 2015 at 10:01

This is possible: http://asciidoctor.org/news/2013/06/03/asciidoclet-announcement/

Reply

Anonymous 3 May 2014 at 09:28

-Xdoclint:none is an invalid option in pre-8 Javadoc

javadoc: error - invalid flag: -Xdoclint:none

Is there a backwards-compatible way to maintain the former behaviour?

Reply

Replies

Ben Hutchison 12 October 2016 at 02:12

I was able to maintain the former behavior in a backwards-compatible way by using a Maven profile activated by the JDK version:

http://www.hastebin.com/iqeyododaw.xml

Tested on JDK 7 and JDK 8.

Reply

Unknown 30 May 2014 at 04:43

I'm so glad they upgraded this stuff to error status. Sure, I had to fix 140 or so cases where it was just a missing caption on a table, but it found 7 legit issues of broken links, incorrect tags, invalid URLs, etc.

Worth it!

Reply

Anonymous 4 June 2014 at 14:12

Seems that the code generated by xjc is not compliant with these new errors.

Reply

Replies

Anonymous 23 July 2014 at 14:53

This! I like the idea of cleaning up our JavaDocs, but I don't want to manually change auto-generated XJC Java files. Since I can't correctly fix the XJC generated Java files the whole thing feels pointless.

Reply

Anonymous 4 July 2014 at 11:56

Also the wsimport tool from JDK8 generates code which is not compatible with this new regime. It seems the tools in JDK8 package have not been tested for doclint compliance.

Reply

Replies

Anonymous 15 September 2014 at 20:12

Yup. Event OpenJDK 8 itself disables doclint for large parts of the build :)

http://mail.openjdk.java.net/pipermail/build-dev/2013-December/011435.html

Reply

Anonymous 7 November 2014 at 10:29

See https://github.com/dropwizard/dropwizard/blob/master/pom.xml#L247-L255
and https://github.com/dropwizard/dropwizard/blob/master/pom.xml#L367-L382

for a backwards compatible way of doing this

Reply

Replies

Anonymous 27 November 2014 at 10:12

Saved my day!

Anonymous 27 April 2015 at 12:12

I doesn't work for jdk7, but at least it works for jdk8

Reply

turbanoff 6 August 2015 at 00:00

Completely disable doclint is a bad advice. Javadoc doclint can be configured to enable/disable some types of checks.
For example to disable all checks except html we can set additionalparam to
-Xdoclint:syntax -Xdoclint:missing -Xdoclint:accessibility -Xdoclint:reference -Xdoclint:syntax
http://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html

Reply

Ramya 25 November 2015 at 06:28

I dont see any restrictions to @default tags in java8. But my build fails saying AnnotationProcessing issue. Are there strict rules for it too, (@defaultpermissions)

Reply

Filip Hanik 2 December 2015 at 16:38

javadoc {
logging.captureStandardError LogLevel.INFO
logging.captureStandardOutput LogLevel.INFO // suppress "## warnings" message
}

Reply

권진 15 February 2016 at 06:27

mvn javadoc:javadoc -Dadditionalparam=-Xdoclint:none

Reply

Srđan Šrepfler 18 April 2016 at 19:32

It's pretty sad this is still not fixed :/ Any Jaxb generated classes break the javadoc linter in JDK8. Get your props together Oracle!

Reply

Unknown 16 August 2016 at 04:45

You shouldn't be running doclint against generated classes anyway. Turn it off for that. Turn warnings off too - you don't get any benefits from those if you can't fix the code anyway.

That said, JAXB does generate some of the worst code known to man, so I wish they would fix their shit too.

But generated code aside, I think these checks are a perfectly good idea for running against code that you're creating yourself. We have found quite a bit of actual invalid markup being generated from ours just from using this tool. And I don't mean the old compiler used to fix it - it would dumbly output the content straight into the output HTML, and result in missing links and the like.

So I think that in general people should turn as many checks on as possible. We run with all lint options turned on and doclint turned on as well.

Reply

Pawandeep Arora 26 August 2016 at 10:24

Thank you Stephen Colebourne. It worked.

Reply

Adym Lincoln 2 October 2016 at 15:22

Very nice! I was struggling with the javadoc in maven...tia, adym

Reply

Anonymous 30 November 2016 at 21:16

Thanks for this...I've been writing Javadoc with embedded tables for years (since 2006). It is ridiculous to have to go back and fix all that pre-existing Javadoc.

This all came to my attention when trying to use an XJC plugin in a Gradle build to generate code and build Javadoc which was working fine under JDK 1.7 and then failed in 1.8

Xdoclint:none to the rescue!

It is silly though. HTML 4.01 is last century! HTML5 would be much better.

Reply

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)