Item 84
git_comments:

1. * * @deprecated * Please use org.apache.commons.lang.StringUtils.isNotBlank() as a replacement

git_commits:

1. **summary:** CLOUDSTACK-10225: Deprecate StringUtils in favor of Apache Commons (#2431)
   message: CLOUDSTACK-10225: Deprecate StringUtils in favor of Apache Commons (#2431) * CLOUDSTACK-10225: Remove unused methods from StringUtils Signed-off-by: Wido den Hollander <wido@widodh.nl> * CLOUDSTACK-10225: Deprecate StringUtils.isNotBlank Signed-off-by: Wido den Hollander <wido@widodh.nl> * CLOUDSTACK-10225: Make isNotBlank a wrapper around Apache Commons Signed-off-by: Wido den Hollander <wido@widodh.nl>

github_issues:

github_issues_comments:

github_pulls:

1. **title:** CLOUDSTACK-10225: Deprecate StringUtils in favor of Apache Commons
   body: We have various places were we use our own StringUtils and other where we use the Apache Commons one. This PR removes unused functions from the StringUtills but it also makes our own isNotBlank() just a wrapper around the Apache Commons. In the future we should remove all references we find to our own StringUtils.isNotBlank()

github_pulls_comments:

1. @wido the travis build tells me you didn't check the noredist build. Let me have a look later today. Thanks for this initiative. also this is more of an improvement than an enhancement, agree?
2. @DaanHoogland Indeed, I always forget about noredist... This is my first (other) babystep in taking this route. The main purpose is to make sure both isNotBlank() functions behave equal.
3. Travis seems happy now @DaanHoogland My IntelliJ was a bit to happy with telling me that method wasn't used :) It's back now.
4. Thanks @rafaelweingartner ! Fixed that comment
5. @blueorangutan package
6. @DaanHoogland a Jenkins job has been kicked to build packages. I'll keep you posted as I make progress.
7. Packaging result: ✔centos6 ✔centos7 ✔debian. JID-1682
8. @blueorangutan test
9. @DaanHoogland a Trillian-Jenkins test job (centos7 mgmt + kvm-centos7) has been kicked to run smoke tests
10. @blueorangutan test
11. @DaanHoogland a Trillian-Jenkins test job (centos7 mgmt + kvm-centos7) has been kicked to run smoke tests
12. @wido @rafaelweingartner I really think we should empty our StringUtils but maintain it as proxy class. That will reduce our 'surface' to the commons version and facilitate any upgrade. We should apply that principle to any library that we use. We are in big trouble now because this principle is not applied to the json library 'gson'. Also with logging we have such very costly upgrade problems. consequently I think we should move all references to commons string utils to our stringutils and make it the sole dependant on the commons package. </rant> merging now
13. I disagree, but that is ok. Cascade/facade and others of the same type to these common libraries will cause more pain than benefits, and in my not so humble opinion will add code without value… There are other patterns one can use to reduce these types of problems you are mentioning. For instance, the logger upgrade, all you need is to change the log configuration, and then change the logger object, right? However, the logging methods will be maintained (info, debug, and so on). That would be easily done if instead of declaring a final static logger object in all classes, we used an inheritance strategy. Therefore, we would reduce greatly the surface of an upgrade like this one. The same applies when using other

libraries that are used in a great number of objects. The problems we have with these upgrade is a consequence of poor design decisions in the past.

14. I may have to revert this! as it turns out I overlooked that no regression test report is included. I just saw a lot of green through flue stricken eyes and merged. As to the logging upgrade, @rafaelweingartner , whether the logging objects are static or instance members doesn't change much for the fact that they need to bee changed in every file. Also in case of a method rename (i.e. debug become deBug (which is a much more accurate camelCase name come to think of it)) We only need to change that in the proxy class and not all over our code. Reducing surface area for external libraries is a best practice in architecture. So is reducing the number of external libraries, which is of course a force opposing the principle of not implementing what is already out there.
15. If we are working with inheritance, I do not see the need to change anything in other classes (children), but the base class. You change the object in the base class, and that will affect automatically all others without needing to change code. Unless some method is removed or change signature, which is not the case in well-known libraries. They will first deprecate, and then only a lot of time later remove it. BTW: debug is already considered a word (I might be totally wrong....). So, there is not camelcase deBug. It feels that it is the same case as `user name` versus `username`
16. @DaanHoogland only to illustrate what I was talking about. I created a minimalist example here: https://github.com/rafaelweingartner/log4j-minimalistic-upgrade-example/commits/master
17. I think you are missing the point @rafaelweingartner. We may *need* to do a major upgrade of a library (log4j/gson/bcprov) or we may need to change the entire library (slf4j and other logging implemetations/StringUtils of several providers). your example just proves a 'simple' upgrade and is actually making my point for me in this line: Having to change protected Logger logger = Logger.getLogger(getClass()); to protected Logger logger = LogManager.getLogger(getClass()); in every 'base'class is not something we want to do. If we create a proxy class it will always be in only one place. this is irrespective of the discussion on static or instance var. It has to do with dealing with dependencies and the issue predates object oriented programming all together.
18. yes, but we should not have 100's of base classes. For instance, in ACS almost everything is extending `ComponentLifecycleBase`. Can you show me a concrete case in ACS that using inheritance like I showed you would not resolve this problem?
19. a single baseclass would actually solve the problem but we do not insert our logger into Object or derive everything from a single base class. also I don't think we should solve the external library interface surface problem with inheritance, a proxy is much easier to read and more maintainable.
20. Automated Test from Blue O
21. I know that we do not do, but that is what I am saying, if we did... we would not only have 1000s of less lines of code initializing logger variables and logger variables with standard names, but also easier upgrade for log implementation. I still do not like the proxy approach, but maybe I need a concrete example.
22. StringUtils is a perfect example. It can hide that we use StringUtil, StringUtilities and StringUtils from org.springframework.util (two versions) org.owasp.esapi (thought we just phased them out we should use those) org.apache.commons.lang and pull in but not use: com.amazonaws.util (from two jars) groovyjarjarantlr com.mysql.jdbc bsh org.codehaus.groovy.util (actually org.eclipse.jetty.util (from two jars) io.netty.util.internal (as the package name show it should not be exposed) org.apache.poi.util In my blatantly inhumble opinion, these should all be hidden behind a single proxy class the cherrypick functionality that is needed for the project (and makes clear which can be removed asap)
23. Ah, now I got your idea... you would like to use all of those libraries. I would do something different... Never use those other libraries (unless, they are unavoidable). I have never seen the need to use them instead of Apache's one (in all of the code I have developed so far...).
24. well, i think the owasp one had real use at some point. and in general some extra might or some reason to change might come along (see the slf4j vs log4j example)
25. yes, I am aware of SLF4J case, but still I would have dealt with it in the same manner said earlier. In these logging implementation case, most of the libraries have similar (not to say the same) method signatures (at least the most used ones). This facilitates the upgrade process in Java, what changes quite considerably is normally the configuration process and their inner working.
26. Trillian test result (tid-2215) Environment: kvm-centos7 (x2), Advanced Networking with Mgmt server 7 Total time taken: -146 seconds Marvin logs: https://github.com/blueorangutan/acs-prs/releases/download/trillian/pr2431-t2215-kvm-centos7.zip Smoke tests completed. 1 look OK, 0 have error(s) Only failed tests results shown below: Test | Result | Time (s) | Test File --- | --- | --- | ---
27. Trillian test result (tid-2216) Environment: kvm-centos7 (x2), Advanced Networking with Mgmt server 7 Total time taken: 48 seconds Marvin logs: https://github.com/blueorangutan/acs-

prs/releases/download/trillian/pr2431-t2216-kvm-centos7.zip Smoke tests completed. 1 look OK, 0 have error(s) Only failed tests results shown below: Test | Result | Time (s) | Test File --- | --- | --- | ---

github_pulls_reviews:

1. this one is used in vmware (noredist)
2. What about referencing `org.apache.commons.lang.StringUtils.IsNotBlank` instead? So, we do not cause confusions.
3. missed that one :(

jira_issues:

jira_issues_comments: