



In the other end of the scale are things like the overhead of maintaining this many modules, and of

course that it's trickier to deal with if you get the modules themselves wrong in the design.

I've found it to be much less overhead than what I'd anticipated, and must say I quite like the finegrained modules. I think that, beyond the architecturally-proper straightjacket, it also helps looking at the code since things are in tiny labelled boxes. All that said, I really don't feel strongly about it, I'd be ok swapping it to a single module (or two) if people feel strongly opposed to strict compartmentalization.



thobe added a note Apr 2, 2015

Owner

I'm very much in favour of small modules for the same reasons. It helps finding code. And it helps finding tests, the test code comes "closer" to the code-under-test, and if you look at for example kernel, finding out where something is tested is extremely hard.



jakewins added a note Apr 2, 2015

Collaborator

The only concern I have is that I, for OCD reasons, would very much like our Maven Central footprint to not contain hundreds of modules. I don't think that's an issue here though, as I expect these modules should not be needed in maven central. I may be wrong there though, but even then we could look into bundling larger modules to maven central to make the public module API easy to digest.



cleishm added a note Apr 2, 2015

Owner

+1 (to both). I like the clear separation of concerns that modules can give. I don't like the plethora of jars that end up on Maven Central and in the product. I don't like the dependency issues that get created, esp. wrt tests.

jakewins added a note Apr 3, 2015

Collaborator

So, on the maven and product issues, I think both can be solved by some well planned packaging (eg. uber-jar for the product and simply not putting this part of the code base on maven central for maven, or uber-jar there as well). The test jars is an issue. Like Tobias notes, its awesome to have the tests close to small distinct modules. However, of course, we often have associated test tooling and utility code which we'd prefer to not duplicate across modules.

IIRC, you sent out a few proposals for options on solving that latter issue just a few days ago :)

Add a line note



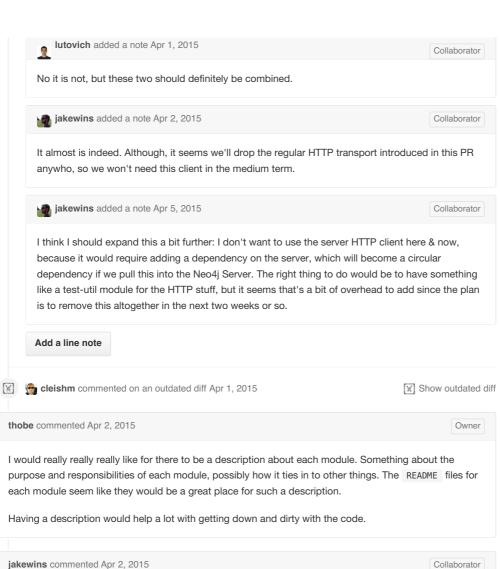
Iutovich commented on the diff Apr 1, 2015

...test/java/org/neo4j/ndp/transport/http/util/HTTP.java View full changes W 36 +import org.neo4j.server.rest.domain.JsonParseException; +import org.neo4j.ndp.messaging.v1.util.BytePrinter; 39 +import static java.util.Collections.unmodifiableMap; +import static org.hamcrest.Matchers.allOf; +import static org.hamcrest.Matchers.anyOf; 41 +import static org.hamcrest.Matchers.containsString; 43 +import static org.hamcrest.Matchers.not; +import static org.junit.Assert.assertThat; +import static org.neo4j.helpers.collection.IteratorUtil.singleOrNull; 46 +import static org.neo4j.helpers.collection.MapUtil.stringMap; +/** 48 + * A tool for performing REST HTTP requests 50 +public class HTTP

lutovich added a note Apr 1, 2015

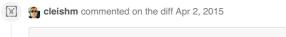
Collaborator

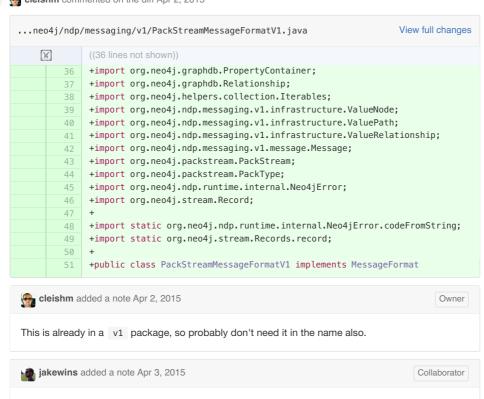
Is it exactly same thing as org.neo4j.test.server.HTTP ?



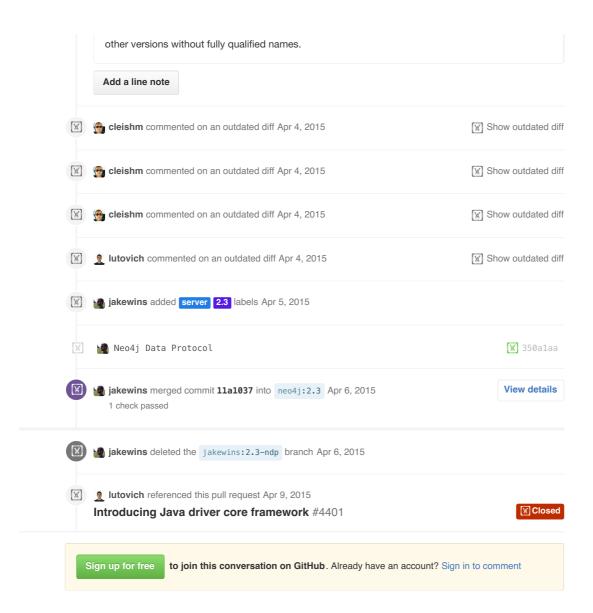


@thobe I've removed the u-ext module for now, and added a README with an overview in the ndp super-module.





I actually like having it in both - the package groups the related stuff for that version, and the version in the name clarifies that this is the v1 message format and allows mixing this class with



© 2015 GitHub, Inc.

nc. Terms Privacy Security Contact

W

Status API Training Shop Blog About Help