Item 114
git_comments:

git_commits:

1. **summary:** CAMEL-14136: Fix ConcurrentModificationException in Scanner (#3313)
   message: CAMEL-14136: Fix ConcurrentModificationException in Scanner (#3313) Internal use of non threadsafe map wrapped with synchronized block, extracted some static patterns into class static block
   label: code-design

github_issues:

github_issues_comments:

github_pulls:

1. **title:** CAMEL-14136: Fix ConcurrentModificationException in Scanner
   body: Internal use of non threadsafe map wrapped with synchronized block, extracted some static patterns into class static block https://issues.apache.org/jira/browse/CAMEL-14136

github_pulls_comments:

1. Some spaces were missing, due to a different formatter in eclipse.

github_pulls_reviews:

jira_issues:

1. **summary:** ConcurrentModificationException when using recipientList with Strings
   description: I already asked this on the mailing list: I encountered a ConcurrentModificationException in Camel: {code:java} java.util.ConcurrentModificationException: null at java.base/java.util.HashMap.computeIfAbsent(HashMap.java:1134) at org.apache.camel.util.Scanner.cachePattern(Scanner.java:305) at org.apache.camel.util.Scanner.<init>(Scanner.java:82) at org.apache.camel.support.ObjectHelper.lambda$createIterable$3(ObjectHelper.java:593) at org.apache.camel.support.ObjectHelper$$Lambda$1118.0000000000000000.iterator(Unknown Source) at org.apache.camel.support.ObjectHelper.createIterator(ObjectHelper.java:435) at org.apache.camel.support.ObjectHelper.createIterator(ObjectHelper.java:412) at org.apache.camel.processor.RecipientList.sendToRecipientList(RecipientList.java:137) at org.apache.camel.processor.RecipientList.process(RecipientList.java:125) at org.apache.camel.processor.Pipeline.doProcess(Pipeline.java:103) at org.apache.camel.processor.Pipeline.lambda$null$2(Pipeline.java:104) at org.apache.camel.processor.Pipeline$$Lambda$1096.0000000000000000.run(Unknown Source) at org.apache.camel.impl.engine.DefaultReactiveExecutor$3.run(DefaultReactiveExecutor.java:116) at org.apache.camel.impl.engine.DefaultReactiveExecutor$Worker.schedule(DefaultReactiveExecutor.java:185) at org.apache.camel.impl.engine.DefaultReactiveExecutor.schedule(DefaultReactiveExecutor.java:67) at org.apache.camel.spi.ReactiveExecutor.schedule(ReactiveExecutor.java:32) at org.apache.camel.processor.MulticastProcessor.lambda$schedule$1(MulticastProcessor.java:249) at org.apache.camel.processor.MulticastProcessor$$Lambda$1099.0000000000000000.run(Unknown Source) at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:515) at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264) at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1128) at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:628) at java.base/java.lang.Thread.run(Thread.java:831) {code} My simplified route is as follows. The input is a list of objects, these are split, and based on the object the recipientList is different. When there are two recipients in the String provided by „getRecipients", the named exception is thrown. Sadly not consistently. {code:java} from("seda:input") .split(method(TestClass.class, "split"), AggregationStrategies.groupedBody()) .parallelProcessing() .recipientList(method(TestClass.class, "getRecipients")) .parallelProcessing() {code} It looks like the

Scanner uses an internal map of compiled Patterns, but this map (LinkedHashMap) is not threadsafe. Response from Claus Ibsen: Yeah its a bug in Camel 3, you are surely welcome to log a JIRA ticket and work on a patch. I can see we (gnodet) introduced a custom fast scanner implementation, but its not thread safe. We could add synchronization block to that compute if absent. And for those constant patterns WHITE_SPACE, FIND_ANY etc we can make them static as java.util.Pattern is thread-safe and create them once in a class static block. I did that and will post a PR-Request to the github project shortly, when the ticket number is known.

jira_issues_comments:

1. Thanks for the PR