Item 91
git_comments:

git_commits:

1. **summary:** GEODE-800: Update fast-classpath-scanner to 2.18.1 (#1430)
   message: GEODE-800: Update fast-classpath-scanner to 2.18.1 (#1430) - This also fixes the issue of picking up nested resources - The new fast-classpath-scanner runs slower than the prior version. This isn't a problem for general use, but some tests have been adjusted to take this into account.

github_issues:

github_issues_comments:

github_pulls:

1. **title:** GEODE-800: Update fast-classpath-scanner to 2.18.1
   body: - This also fixes the issue of picking up nested resources Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [x] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [x] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [x] Is your initial contribution a single, squashed commit? - [x] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, you check travis-ci for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
2. **title:** GEODE-800: Update fast-classpath-scanner to 2.18.1
   body: - This also fixes the issue of picking up nested resources Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [x] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [x] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [x] Is your initial contribution a single, squashed commit? - [x] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, you check travis-ci for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.
3. **title:** GEODE-800: Update fast-classpath-scanner to 2.18.1
   body: - This also fixes the issue of picking up nested resources Thank you for submitting a contribution to Apache Geode. In order to streamline the review of the contribution we ask you to ensure the following steps have been taken: ### For all changes: - [x] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [x] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [x] Is your initial contribution a single, squashed commit? - [x] Does `gradlew build` run cleanly? - [ ] Have you written or updated unit tests to verify your changes? - [ ] If adding new dependencies to the code, are these dependencies licensed in a way that is compatible for inclusion under [ASF 2.0](http://www.apache.org/legal/resolved.html#category-a)? ### Note: Please ensure that once the PR is submitted, you check travis-ci for build issues and submit an update to your PR as soon as possible. If you need help, please send an email to dev@geode.apache.org.

github_pulls_comments:

1. precheckin is green

github_pulls_reviews:

jira_issues:

1. **summary:** Geode's classloading mechanism is unable to resolve classes found within nested jars
   description: This issue is particularly evident when using Geode in a Spring Boot app which creates an 'über' jar containing all dependent jars. When Geode is launched in this context, the following errors can be seen: {noformat} [warn 2016/01/20 08:53:29.431 PST <main> tid=0xd] (tid=13 msgId=0) Required Commands classes were not loaded. Check logs for errors. java.lang.IllegalStateException: Required Commands classes were not loaded. Check logs for errors. at com.gemstone.gemfire.management.internal.cli.CommandManager.raiseExceptionIfEmpty(CommandManager.java:249) at com.gemstone.gemfire.management.internal.cli.CommandManager.loadCommands(CommandManager.java:188) at com.gemstone.gemfire.management.internal.cli.CommandManager.<init>(CommandManager.java:86) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:278) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:258) at com.gemstone.gemfire.management.internal.cli.remote.CommandProcessor.<init>(CommandProcessor.java:58) ...

{noformat} The problem here is in {{ClasspathScanLoadHelper.getClasses()}}. In this method we call: {noformat} Enumeration<URL> resources = ClassPathLoader.getLatest().getResources(packagePath); {noformat} However {{getResources()}} doesn't just work against the 'latest' classloader, but also considers the thread context classloader. In the case of a Spring Boot app, Spring does provide such a classloader and {{getResources}} is able to find the necessary resources {{CommandMarker}} classes. (These classes are found within a nested jar. For ex. {{jar:file:/Users/jdeppe/src/woddrive/WodDrive-GF-Server/target/WodDriveGFServer.jar!/lib/gemfire-core-1.0.0-incubating-SNAPSHOT.jar!/com/gemstone/gemfire/management/internal/cli/commands}}). This is all fine, but subsequent code doesn't consider classes (or packages) within nested jars, and in addition, when classes actually get resolved, the thread context classloader (where those resources might have come from) is not considered.

2. **summary:** Geode's classloading mechanism is unable to resolve classes found within nested jars
   description: This issue is particularly evident when using Geode in a Spring Boot app which creates an 'über' jar containing all dependent jars. When Geode is launched in this context, the following errors can be seen: {noformat} [warn 2016/01/20 08:53:29.431 PST <main> tid=0xd] (tid=13 msgId=0) Required Commands classes were not loaded. Check logs for errors. java.lang.IllegalStateException: Required Commands classes were not loaded. Check logs for errors. at com.gemstone.gemfire.management.internal.cli.CommandManager.raiseExceptionIfEmpty(CommandManager.java:249) at com.gemstone.gemfire.management.internal.cli.CommandManager.loadCommands(CommandManager.java:188) at com.gemstone.gemfire.management.internal.cli.CommandManager.<init>(CommandManager.java:86) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:278) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:258) at com.gemstone.gemfire.management.internal.cli.remote.CommandProcessor.<init>(CommandProcessor.java:58) ... {noformat} The problem here is in {{ClasspathScanLoadHelper.getClasses()}}. In this method we call: {noformat} Enumeration<URL> resources = ClassPathLoader.getLatest().getResources(packagePath); {noformat} However {{getResources()}} doesn't just work against the 'latest' classloader, but also considers the thread context classloader. In the case of a Spring Boot app, Spring does provide such a classloader and {{getResources}} is able to find the necessary resources {{CommandMarker}} classes. (These classes are found within a nested jar. For ex. {{jar:file:/Users/jdeppe/src/woddrive/WodDrive-GF-Server/target/WodDriveGFServer.jar!/lib/gemfire-core-1.0.0-incubating-SNAPSHOT.jar!/com/gemstone/gemfire/management/internal/cli/commands}}). This is all fine, but subsequent code doesn't consider classes (or packages) within nested jars, and in addition, when classes actually get resolved, the thread context classloader (where those resources might have come from) is not considered.
3. **summary:** Geode's classloading mechanism is unable to resolve classes found within nested jars
   description: This issue is particularly evident when using Geode in a Spring Boot app which creates an 'über' jar containing all dependent jars. When Geode is launched in this context, the following errors can be seen: {noformat} [warn 2016/01/20 08:53:29.431 PST <main> tid=0xd] (tid=13 msgId=0) Required Commands classes were not loaded. Check logs for errors. java.lang.IllegalStateException: Required Commands classes were not loaded. Check logs for errors. at com.gemstone.gemfire.management.internal.cli.CommandManager.raiseExceptionIfEmpty(CommandManager.java:249) at com.gemstone.gemfire.management.internal.cli.CommandManager.loadCommands(CommandManager.java:188) at com.gemstone.gemfire.management.internal.cli.CommandManager.<init>(CommandManager.java:86) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:278) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:258) at com.gemstone.gemfire.management.internal.cli.remote.CommandProcessor.<init>(CommandProcessor.java:58) ... {noformat} The problem here is in {{ClasspathScanLoadHelper.getClasses()}}. In this method we call: {noformat} Enumeration<URL> resources = ClassPathLoader.getLatest().getResources(packagePath); {noformat} However {{getResources()}} doesn't just work against the 'latest' classloader, but also considers the thread context classloader. In the case of a Spring Boot app, Spring does provide such a classloader and {{getResources}} is able to find the necessary resources {{CommandMarker}} classes. (These classes are found within a nested jar. For ex. {{jar:file:/Users/jdeppe/src/woddrive/WodDrive-GF-Server/target/WodDriveGFServer.jar!/lib/gemfire-core-1.0.0-incubating-SNAPSHOT.jar!/com/gemstone/gemfire/management/internal/cli/commands}}). This is all fine, but subsequent code doesn't consider classes (or packages) within nested jars, and in addition, when classes actually get resolved, the thread context classloader (where those resources might have come from) is not considered.
   label: documentation
4. **summary:** Geode's classloading mechanism is unable to resolve classes found within nested jars
   description: This issue is particularly evident when using Geode in a Spring Boot app which creates an 'über' jar containing all dependent jars. When Geode is launched in this context, the following errors can be seen: {noformat} [warn 2016/01/20 08:53:29.431 PST <main> tid=0xd] (tid=13 msgId=0) Required Commands classes were not loaded. Check logs for errors. java.lang.IllegalStateException: Required Commands classes were not loaded. Check logs for errors. at com.gemstone.gemfire.management.internal.cli.CommandManager.raiseExceptionIfEmpty(CommandManager.java:249) at com.gemstone.gemfire.management.internal.cli.CommandManager.loadCommands(CommandManager.java:188) at com.gemstone.gemfire.management.internal.cli.CommandManager.<init>(CommandManager.java:86) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:278) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:258) at com.gemstone.gemfire.management.internal.cli.remote.CommandProcessor.<init>(CommandProcessor.java:58) ... {noformat} The problem here is in {{ClasspathScanLoadHelper.getClasses()}}. In this method we call: {noformat} Enumeration<URL> resources = ClassPathLoader.getLatest().getResources(packagePath); {noformat} However

{{getResources()}} doesn't just work against the 'latest' classloader, but also considers the thread context classloader. In the case of a Spring Boot app, Spring does provide such a classloader and {{getResources}} is able to find the necessary resources {{CommandMarker}} classes. (These classes are found within a nested jar. For ex. {{jar:file:/Users/jdeppe/src/woddrive/WodDrive-GF-Server/target/WodDriveGFServer.jar!/lib/gemfire-core-1.0.0-incubating-SNAPSHOT.jar!/com/gemstone/gemfire/management/internal/cli/commands}}). This is all fine, but subsequent code doesn't consider classes (or packages) within nested jars, and in addition, when classes actually get resolved, the thread context classloader (where those resources might have come from) is not considered.

5. **summary:** Geode's classloading mechanism is unable to resolve classes found within nested jars
   description: This issue is particularly evident when using Geode in a Spring Boot app which creates an 'über' jar containing all dependent jars. When Geode is launched in this context, the following errors can be seen: {noformat} [warn 2016/01/20 08:53:29.431 PST <main> tid=0xd] (tid=13 msgId=0) Required Commands classes were not loaded. Check logs for errors. java.lang.IllegalStateException: Required Commands classes were not loaded. Check logs for errors. at com.gemstone.gemfire.management.internal.cli.CommandManager.raiseExceptionIfEmpty(CommandManager.java:249) at com.gemstone.gemfire.management.internal.cli.CommandManager.loadCommands(CommandManager.java:188) at com.gemstone.gemfire.management.internal.cli.CommandManager.<init>(CommandManager.java:86) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:278) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:258) at com.gemstone.gemfire.management.internal.cli.remote.CommandProcessor.<init>(CommandProcessor.java:58) ... {noformat} The problem here is in {{ClasspathScanLoadHelper.getClasses()}}. In this method we call: {noformat} Enumeration<URL> resources = ClassPathLoader.getLatest().getResources(packagePath); {noformat} However {{getResources()}} doesn't just work against the 'latest' classloader, but also considers the thread context classloader. In the case of a Spring Boot app, Spring does provide such a classloader and {{getResources}} is able to find the necessary resources {{CommandMarker}} classes. (These classes are found within a nested jar. For ex. {{jar:file:/Users/jdeppe/src/woddrive/WodDrive-GF-Server/target/WodDriveGFServer.jar!/lib/gemfire-core-1.0.0-incubating-SNAPSHOT.jar!/com/gemstone/gemfire/management/internal/cli/commands}}). This is all fine, but subsequent code doesn't consider classes (or packages) within nested jars, and in addition, when classes actually get resolved, the thread context classloader (where those resources might have come from) is not considered.
6. **summary:** Geode's classloading mechanism is unable to resolve classes found within nested jars
   description: This issue is particularly evident when using Geode in a Spring Boot app which creates an 'über' jar containing all dependent jars. When Geode is launched in this context, the following errors can be seen: {noformat} [warn 2016/01/20 08:53:29.431 PST <main> tid=0xd] (tid=13 msgId=0) Required Commands classes were not loaded. Check logs for errors. java.lang.IllegalStateException: Required Commands classes were not loaded. Check logs for errors. at com.gemstone.gemfire.management.internal.cli.CommandManager.raiseExceptionIfEmpty(CommandManager.java:249) at com.gemstone.gemfire.management.internal.cli.CommandManager.loadCommands(CommandManager.java:188) at com.gemstone.gemfire.management.internal.cli.CommandManager.<init>(CommandManager.java:86) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:278) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:258) at com.gemstone.gemfire.management.internal.cli.remote.CommandProcessor.<init>(CommandProcessor.java:58) ... {noformat} The problem here is in {{ClasspathScanLoadHelper.getClasses()}}. In this method we call: {noformat} Enumeration<URL> resources = ClassPathLoader.getLatest().getResources(packagePath); {noformat} However {{getResources()}} doesn't just work against the 'latest' classloader, but also considers the thread context classloader. In the case of a Spring Boot app, Spring does provide such a classloader and {{getResources}} is able to find the necessary resources {{CommandMarker}} classes. (These classes are found within a nested jar. For ex. {{jar:file:/Users/jdeppe/src/woddrive/WodDrive-GF-Server/target/WodDriveGFServer.jar!/lib/gemfire-core-1.0.0-incubating-SNAPSHOT.jar!/com/gemstone/gemfire/management/internal/cli/commands}}). This is all fine, but subsequent code doesn't consider classes (or packages) within nested jars, and in addition, when classes actually get resolved, the thread context classloader (where those resources might have come from) is not considered.
7. **summary:** Geode's classloading mechanism is unable to resolve classes found within nested jars
   description: This issue is particularly evident when using Geode in a Spring Boot app which creates an 'über' jar containing all dependent jars. When Geode is launched in this context, the following errors can be seen: {noformat} [warn 2016/01/20 08:53:29.431 PST <main> tid=0xd] (tid=13 msgId=0) Required Commands classes were not loaded. Check logs for errors. java.lang.IllegalStateException: Required Commands classes were not loaded. Check logs for errors. at com.gemstone.gemfire.management.internal.cli.CommandManager.raiseExceptionIfEmpty(CommandManager.java:249) at com.gemstone.gemfire.management.internal.cli.CommandManager.loadCommands(CommandManager.java:188) at com.gemstone.gemfire.management.internal.cli.CommandManager.<init>(CommandManager.java:86) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:278) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:258) at com.gemstone.gemfire.management.internal.cli.remote.CommandProcessor.<init>(CommandProcessor.java:58) ... {noformat} The problem here is in {{ClasspathScanLoadHelper.getClasses()}}. In this method we call: {noformat} Enumeration<URL> resources = ClassPathLoader.getLatest().getResources(packagePath); {noformat} However {{getResources()}} doesn't just work against the 'latest' classloader, but also considers the thread context classloader. In the case of a Spring Boot app, Spring does provide such a classloader and {{getResources}} is able to find the necessary resources {{CommandMarker}} classes. (These classes are found within a nested jar. For ex.

{{jar:file:/Users/jdeppe/src/woddrive/WodDrive-GF-Server/target/WodDriveGFServer.jar!/lib/gemfire-core-1.0.0-incubating-SNAPSHOT.jar!/com/gemstone/gemfire/management/internal/cli/commands}}). This is all fine, but subsequent code doesn't consider classes (or packages) within nested jars, and in addition, when classes actually get resolved, the thread context classloader (where those resources might have come from) is not considered.

8. **summary:** Geode's classloading mechanism is unable to resolve classes found within nested jars
   description: This issue is particularly evident when using Geode in a Spring Boot app which creates an 'über' jar containing all dependent jars. When Geode is launched in this context, the following errors can be seen: {noformat} [warn 2016/01/20 08:53:29.431 PST <main> tid=0xd] (tid=13 msgId=0) Required Commands classes were not loaded. Check logs for errors. java.lang.IllegalStateException: Required Commands classes were not loaded. Check logs for errors. at com.gemstone.gemfire.management.internal.cli.CommandManager.raiseExceptionIfEmpty(CommandManager.java:249) at com.gemstone.gemfire.management.internal.cli.CommandManager.loadCommands(CommandManager.java:188) at com.gemstone.gemfire.management.internal.cli.CommandManager.<init>(CommandManager.java:86) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:278) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:258) at com.gemstone.gemfire.management.internal.cli.remote.CommandProcessor.<init>(CommandProcessor.java:58) ... {noformat} The problem here is in {{ClasspathScanLoadHelper.getClasses()}}. In this method we call: {noformat} Enumeration<URL> resources = ClassPathLoader.getLatest().getResources(packagePath); {noformat} However {{getResources()}} doesn't just work against the 'latest' classloader, but also considers the thread context classloader. In the case of a Spring Boot app, Spring does provide such a classloader and {{getResources}} is able to find the necessary resources {{CommandMarker}} classes. (These classes are found within a nested jar. For ex. {{jar:file:/Users/jdeppe/src/woddrive/WodDrive-GF-Server/target/WodDriveGFServer.jar!/lib/gemfire-core-1.0.0-incubating-SNAPSHOT.jar!/com/gemstone/gemfire/management/internal/cli/commands}}). This is all fine, but subsequent code doesn't consider classes (or packages) within nested jars, and in addition, when classes actually get resolved, the thread context classloader (where those resources might have come from) is not considered.
9. **summary:** Geode's classloading mechanism is unable to resolve classes found within nested jars
   description: This issue is particularly evident when using Geode in a Spring Boot app which creates an 'über' jar containing all dependent jars. When Geode is launched in this context, the following errors can be seen: {noformat} [warn 2016/01/20 08:53:29.431 PST <main> tid=0xd] (tid=13 msgId=0) Required Commands classes were not loaded. Check logs for errors. java.lang.IllegalStateException: Required Commands classes were not loaded. Check logs for errors. at com.gemstone.gemfire.management.internal.cli.CommandManager.raiseExceptionIfEmpty(CommandManager.java:249) at com.gemstone.gemfire.management.internal.cli.CommandManager.loadCommands(CommandManager.java:188) at com.gemstone.gemfire.management.internal.cli.CommandManager.<init>(CommandManager.java:86) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:278) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:258) at com.gemstone.gemfire.management.internal.cli.remote.CommandProcessor.<init>(CommandProcessor.java:58) ... {noformat} The problem here is in {{ClasspathScanLoadHelper.getClasses()}}. In this method we call: {noformat} Enumeration<URL> resources = ClassPathLoader.getLatest().getResources(packagePath); {noformat} However {{getResources()}} doesn't just work against the 'latest' classloader, but also considers the thread context classloader. In the case of a Spring Boot app, Spring does provide such a classloader and {{getResources}} is able to find the necessary resources {{CommandMarker}} classes. (These classes are found within a nested jar. For ex. {{jar:file:/Users/jdeppe/src/woddrive/WodDrive-GF-Server/target/WodDriveGFServer.jar!/lib/gemfire-core-1.0.0-incubating-SNAPSHOT.jar!/com/gemstone/gemfire/management/internal/cli/commands}}). This is all fine, but subsequent code doesn't consider classes (or packages) within nested jars, and in addition, when classes actually get resolved, the thread context classloader (where those resources might have come from) is not considered.
10. **summary:** Geode's classloading mechanism is unable to resolve classes found within nested jars
    description: This issue is particularly evident when using Geode in a Spring Boot app which creates an 'über' jar containing all dependent jars. When Geode is launched in this context, the following errors can be seen: {noformat} [warn 2016/01/20 08:53:29.431 PST <main> tid=0xd] (tid=13 msgId=0) Required Commands classes were not loaded. Check logs for errors. java.lang.IllegalStateException: Required Commands classes were not loaded. Check logs for errors. at com.gemstone.gemfire.management.internal.cli.CommandManager.raiseExceptionIfEmpty(CommandManager.java:249) at com.gemstone.gemfire.management.internal.cli.CommandManager.loadCommands(CommandManager.java:188) at com.gemstone.gemfire.management.internal.cli.CommandManager.<init>(CommandManager.java:86) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:278) at com.gemstone.gemfire.management.internal.cli.CommandManager.getInstance(CommandManager.java:258) at com.gemstone.gemfire.management.internal.cli.remote.CommandProcessor.<init>(CommandProcessor.java:58) ... {noformat} The problem here is in {{ClasspathScanLoadHelper.getClasses()}}. In this method we call: {noformat} Enumeration<URL> resources = ClassPathLoader.getLatest().getResources(packagePath); {noformat} However {{getResources()}} doesn't just work against the 'latest' classloader, but also considers the thread context classloader. In the case of a Spring Boot app, Spring does provide such a classloader and {{getResources}} is able to find the necessary resources {{CommandMarker}} classes. (These classes are found within a nested jar. For ex. {{jar:file:/Users/jdeppe/src/woddrive/WodDrive-GF-Server/target/WodDriveGFServer.jar!/lib/gemfire-core-1.0.0-incubating-SNAPSHOT.jar!/com/gemstone/gemfire/management/internal/cli/commands}}). This is all fine, but

subsequent code doesn't consider classes (or packages) within nested jars, and in addition, when classes actually get resolved, the thread context classloader (where those resources might have come from) is not considered.

jira_issues_comments:

1. The problem seems to disappear if You override the classloaders in the FastClasspathScanner, e.g. this way (class is org.apache.geode.management.internal.cli.util.ClasspathScanLoadHelper): {code:java} public static Set<Class<?>> scanPackageForClassesImplementing(String packageToScan, Class<?> implementedInterface) { Set<Class<?>> classesImplementing = new HashSet<>(); new FastClasspathScanner(packageToScan) .overrideClassLoaders(ClasspathScanLoadHelper.class.getClassLoader()) .matchClassesImplementing(implementedInterface, classesImplementing::add).scan(); return classesImplementing.stream().filter(ClasspathScanLoadHelper::isInstantiable) .collect(toSet()); } {code}
2. What problem exactly are you seeing (and how could I reproduce it) that gets solved by your suggestion? Just to be clear, that {{FastClsspathScanner.overrideClassLoaders}} call is interrogating the provided classloader for it's *current classpath elements*. It is never used to actually resolve any classes.
3. **body:** Hello Jens, I've attached a test case for the problem we are facing. It contains a readme.md in which I try to explain the problem and its symptoms. Please let me know if I can be of any help.
   label: documentation
4. Hey [~francesco.foresti], The issue can be resolved by using the latest version of the {{fast-classpath-scanner}} library as the dependency, namely {{2.18.1}}, instead of the one transitively included by Geode, namely {{2.0.11}}. {quote} The only remaining question is : why Geode doesn't depend on the correct version by default?. {quote} Geode proactively updates its dependencies regularly (there's actually a [pull request|https://github.com/apache/geode/pull/1400] opened to update some dependencies at the moment) but, as you can see from the [Maven Central Repository|https://mvnrepository.com/artifact/io.github.lukehutch/fast-classpath-scanner] and the relevant [Source Code Releases|https://github.com/lukehutch/fast-classpath-scanner/releases], this particular library has itself changed regularly these last couple of weeks. Hope this helps. Cheers.
5. Commit 68deb0db9578434e9043d56cc48ae1fae4166c5d in geode's branch refs/heads/develop from [~jens.deppe] [https://gitbox.apache.org/repos/asf?p=geode.git;h=68deb0d] GEODE-800: Update fast-classpath-scanner to 2.18.1 (#1430) - This also fixes the issue of picking up nested resources - The new fast-classpath-scanner runs slower than the prior version. This isn't a problem for general use, but some tests have been adjusted to take this into account.
6. Commit 68deb0db9578434e9043d56cc48ae1fae4166c5d in geode's branch refs/heads/feature/GEODE-3967 from [~jens.deppe] [https://gitbox.apache.org/repos/asf?p=geode.git;h=68deb0d] GEODE-800: Update fast-classpath-scanner to 2.18.1 (#1430) - This also fixes the issue of picking up nested resources - The new fast-classpath-scanner runs slower than the prior version. This isn't a problem for general use, but some tests have been adjusted to take this into account.
7. Commit 53178f8e375c05cf093723756a34f5e363e4b63d in geode's branch refs/heads/develop from [~jens.deppe] [https://gitbox.apache.org/repos/asf?p=geode.git;h=53178f8] GEODE-800: Reduce test iterations to avoid timeouts
8. Commit 53178f8e375c05cf093723756a34f5e363e4b63d in geode's branch refs/heads/feature/GEODE-3967 from [~jens.deppe] [https://gitbox.apache.org/repos/asf?p=geode.git;h=53178f8] GEODE-800: Reduce test iterations to avoid timeouts