Item 285
git_comments:

1. Skip some of the Converters from Spring Shell for our customization

git_commits:

1. **summary:** GEODE-3007: Simplify support for custom GFSH commands (#1042)
   message: GEODE-3007: Simplify support for custom GFSH commands (#1042) * GEODE-3007: Simplify support for custom GFSH commands

github_issues:

github_issues_comments:

github_pulls:

1. **title:** GEODE-3007: Simplify support for custom GFSH commands
   body: 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? - [x] 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:

github_pulls_reviews:

1. All uses of this property have been removed in this change - is that going to be a breaking change anywhere?
2. Surely the logic in these nested foreach's could be simplified.
3. At the very least, that should be a `Set` and not a `HashSet` on the left. If we wanted to collapse this into a `stream`, we could. ``` Set<Converter<?>> converters = new HashSet<>(); for (Converter c : commandManager.getConverters()) { converters.add(c); } ``` It seems to me that this is just doing some strange workaround to get a set of `Converter` to a set of `Converter<?>`. That could be rewritten as ``` Set<Converter<?>> converters = commandManager.getConverters().stream().map(c -> (Converter<?>) c).collect(Collectors.toSet()); ``` But at that point, the whole loop would go from ``` for (Converter converter : commandManager.getConverters()) { if (converter.getClass().isAssignableFrom(ArrayConverter.class)) { ArrayConverter arrayConverter = (ArrayConverter) converter; Set<Converter<?>> converters = commandManager.getConverters().stream().map(c -> (Converter<?>) c).collect(Collectors.toSet()); arrayConverter.setConverters(converters); } add(converter); } ``` to ``` commandManager.getConverters() .stream() .filter(converter -> converter.getClass().isAssignableFrom(ArrayConverter.class)) .forEach(converter -> ((ArrayConverter) converter).setConverters(commandManager.getConverters().stream().map(c -> (Converter<?>) c).collect(Collectors.toSet()))); commandManager.getConverters().forEach(this::add); ``` with the caveat that the `add(converter)` is now happening for each after the others have been addressed, and not in the middle of the loop. But I don't think that should really matter, should it..? Looking at that, I don't really believe it to be more readable than the first code block in this comment. So probably split the difference, with ``` for (Converter converter : commandManager.getConverters()) { if (converter.getClass().isAssignableFrom(ArrayConverter.class)) { ArrayConverter arrayConverter = (ArrayConverter) converter; arrayConverter.setConverters(commandManager.getConverters().stream() .map(c -> (Converter<?>) c).collect(Collectors.toSet())); } add(converter); } ``` I don't know. I guess you could one step farther and reduce to ``` for (Converter converter : commandManager.getConverters()) { if (converter.getClass().isAssignableFrom(ArrayConverter.class)) { ((ArrayConverter) converter).setConverters(commandManager.getConverters().stream() .map(c -> (Converter<?>) c).collect(Collectors.toSet())); } add(converter); } ``` but I personally don't like one-lining casts and method calls. ... This comment got a lot bigger than I originally intended...

4. Also, just for the record: all of the above is predicated on a hazy understanding of the difference between `Converter` and `Converter<?>`.
5. The need here appears to stem from the way generics are being used (possibly incorrectly). Ideally we should just be able to say `HashSet<Converter<?>> converters = new HashSet<> (commandManager.getConverters();`. However that doesn't work because `getConverters` returns a `Set<Converter>` and not a `Set<Converter<?>>`. Ultimately the `setConverter` call wants to take a `Set<Converter<?>>` as an argument. Hence the need for this inner loop to keep the compiler happy.
6. No, because the property itself isn't gone.

jira_issues:

1. **summary:** Simplify support for custom GFSH commands
   description: Geode currently supports three ways to load GFSH commands: 1. Scan the classpath for commands in "org.apache.geode.management.internal.cli.commands" 2. Scan the classpath for commands in a package specified by a user via the "user-command-packages" system property. 3. Scan the classpath for commands registered in files inside META-INF.services (e.g. "geode-core/src/test/resources/META-INF/services/org.springframework.shell.core.CommandMarker") After the improvements made by GEODE-2989, there is no reason to require a user to specify the location of their custom commands via one of these mechanisms. Instead, we should simply scan the entire classpath for any classes implementing CommandMarker (regardless of whatever packages they live in).

jira_issues_comments:

1. jaredjstewart opened a new pull request #1011: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1011 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? - [x] 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. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
2. jinmeiliao commented on a change in pull request #1011: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1011#discussion_r148830955 ########## File path: geode-core/src/main/java/org/apache/geode/distributed/ConfigurationProperties.java ########## @@ -1849,19 +1849,7 @@ * <U>Since</U>: GemFire 8.0 */ String USE_CLUSTER_CONFIGURATION = "use-cluster-configuration"; - /** - * The static String definition of the "user-command-packages" property <a - * name="user-command-packages"/a> - * </p> - * <U>Description</U>: A comma separated list of Java packages that contain classes implementing - * the <code>CommandMarker</code> interface. Matching classes will be loaded when the VM starts - * and will be available in the GFSH command-line utility. - * </p> - * <U>Default</U>: <code>""</code> - * </p> - * <U>Since</U>: GemFire 8.0 - */ - String USER_COMMAND_PACKAGES = "user-command-packages"; Review comment: since this is public. I wonder if we should deprecate it first, but have it as a no-effect config property. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
3. PurelyApplied commented on a change in pull request #1011: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1011#discussion_r148836689 ########## File path: geode-assembly/build.gradle ########## @@ -191,12 +193,14 @@ def cp = { it.contains('log4j-slf4j-impl') || it.contains('shiro') || it.contains('slf4j-api') || + it.contains('spring-beans') || + it.contains('spring-context') || + it.contains('spring-context-support') || it.contains('spring-core') || it.contains('spring-shell') || it.contains('snappy') || it.contains('jgroups') || it.contains('netty') || - Review comment: If we're pruning this space, we should prune line 210, too, for consistency. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

4. PurelyApplied commented on a change in pull request #1011: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1011#discussion_r148836195 ########## File path: geode-core/build.gradle ########## @@ -111,10 +112,8 @@ dependencies { exclude module: 'aopalliance' exclude module: 'asm' exclude module: 'cglib' - exclude module: 'guava' exclude module: 'spring-aop' - exclude module: 'spring-context-support' - ext.optional = true + ext.optional = true Review comment: Also a nit-pick, but `spotless` doesn't examine the build files. This needs to tab'd in. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
5. PurelyApplied commented on a change in pull request #1011: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1011#discussion_r148835835 ########## File path: geode-assembly/src/test/java/org/apache/geode/management/internal/cli/commands/StatusLocatorRealGfshTest.java ########## @@ -30,7 +30,6 @@ @Test Review comment: Even GitHub knows I'm being ridiculous and won't let me tag up in line 23, where you didn't do anything... But as long as we're in here, imports should be alphabetized. Line 23's `import org.apache.geode.test.junit.categories.AcceptanceTest` should be at the top of the `org.apache` block. Also, `geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessor.java` and `geode-core/src/test/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessorTest.java` have unused imports. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
6. PurelyApplied commented on a change in pull request #1011: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1011#discussion_r148836503 ########## File path: geode-assembly/build.gradle ########## @@ -160,12 +160,14 @@ def cp = { // depedencies from geode-core Review comment: The nits keep rolling: `depedencies` -> `dependencies` ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
7. PurelyApplied commented on a change in pull request #1011: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1011#discussion_r148838062 ########## File path: geode-core/src/main/java/org/apache/geode/distributed/ConfigurationProperties.java ########## @@ -1849,19 +1849,7 @@ * <U>Since</U>: GemFire 8.0 */ String USE_CLUSTER_CONFIGURATION = "use-cluster-configuration"; - /** - * The static String definition of the "user-command-packages" property <a - * name="user-command-packages"/a> - * </p> - * <U>Description</U>: A comma separated list of Java packages that contain classes implementing - * the <code>CommandMarker</code> interface. Matching classes will be loaded when the VM starts - * and will be available in the GFSH command-line utility. - * </p> - * <U>Default</U>: <code>""</code> - * </p> - * <U>Since</U>: GemFire 8.0 - */ - String USER_COMMAND_PACKAGES = "user-command-packages"; + Review comment: Drop the carriage return for consistency within the file. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
8. PurelyApplied commented on a change in pull request #1011: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1011#discussion_r148837761 ########## File path: geode-core/src/test/java/org/apache/geode/management/internal/cli/CommandManagerJUnitTest.java ########## @@ -45,46 +37,6 @@ public class CommandManagerJUnitTest { private static final String COMMAND1_NAME = "command1"; Review comment: I love seeing the purge, but should `COMMAND1_NAME` not also be on that chop block? ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
9. jdeppe-pivotal commented on a change in pull request #1011: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1011#discussion_r148852585 ########## File path: geode-core/src/main/java/org/apache/geode/management/internal/cli/CommandManager.java ########## @@ -14,293 +14,96 @@ */ package org.apache.geode.management.internal.cli; -import static org.apache.geode.distributed.ConfigurationProperties.USER_COMMAND_PACKAGES; -import java.lang.reflect.Method; -import java.util.ArrayList; -import java.util.HashSet; -import java.util.Iterator; -

import java.util.List; -import java.util.Properties; -import java.util.ServiceConfigurationError; -import java.util.ServiceLoader; +import static java.util.stream.Collectors.toSet; + +import java.util.Objects; import java.util.Set; -import java.util.StringTokenizer; +import java.util.function.Predicate; +import com.google.common.collect.ImmutableSet; Review comment: It seems this is the only reason we're pulling in guava? I'd prefer not to pull in a new dependency if we don't need to (especially when it is not being used pervasively). It reduces the chance of future conflicts with user's application code which also uses that dependency. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

10. metatype commented on a change in pull request #1011: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1011#discussion_r148854953 ########## File path: geode-assembly/build.gradle ########## @@ -193,19 +195,20 @@ def cp = { it.contains('log4j-slf4j-impl') || it.contains('shiro') || it.contains('slf4j-api') || + it.contains('spring-beans') || Review comment: This might cause a conflict with jars on the `gemfire-web` classpath. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
11. metatype commented on a change in pull request #1011: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1011#discussion_r148855312 ########## File path: geode-core/src/main/java/org/apache/geode/management/internal/cli/CommandManager.java ########## @@ -14,293 +14,96 @@ */ package org.apache.geode.management.internal.cli; -import static org.apache.geode.distributed.ConfigurationProperties.USER_COMMAND_PACKAGES; -import java.lang.reflect.Method; -import java.util.ArrayList; -import java.util.HashSet; -import java.util.Iterator; -import java.util.List; -import java.util.Properties; -import java.util.ServiceConfigurationError; -import java.util.ServiceLoader; +import static java.util.stream.Collectors.toSet; + +import java.util.Objects; import java.util.Set; -import java.util.StringTokenizer; +import java.util.function.Predicate; +import com.google.common.collect.ImmutableSet; Review comment: +1 ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
12. jaredjstewart commented on a change in pull request #1011: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1011#discussion_r148888154 ########## File path: geode-assembly/build.gradle ########## @@ -193,19 +195,20 @@ def cp = { it.contains('log4j-slf4j-impl') || it.contains('shiro') || it.contains('slf4j-api') || + it.contains('spring-beans') || Review comment: The reason this was added is that `CommandManager` will now try to load any classes implementing `CommandMarker`, rather than only those in the `org.apache.geode.management.internal.cli.commands` package. This results in the CommandManager trying recognizing the built-in `HelpCommands` command from spring-shell, and that class has a dependency on spring-beans. We could probably exclude classes by name rather than by an actual reference to the `Class` to avoid loading blacklisted commands (and thus remove the need for this dependency). ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
13. jaredjstewart commented on a change in pull request #1011: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1011#discussion_r148888175 ########## File path: geode-core/src/main/java/org/apache/geode/management/internal/cli/CommandManager.java ########## @@ -14,293 +14,96 @@ */ package org.apache.geode.management.internal.cli; -import static org.apache.geode.distributed.ConfigurationProperties.USER_COMMAND_PACKAGES; -import java.lang.reflect.Method; -import java.util.ArrayList; -import java.util.HashSet; -import java.util.Iterator; -import java.util.List; -import java.util.Properties; -import java.util.ServiceConfigurationError; -import java.util.ServiceLoader; +import static java.util.stream.Collectors.toSet; + +import java.util.Objects; import java.util.Set; -import java.util.StringTokenizer; +import java.util.function.Predicate; +import com.google.common.collect.ImmutableSet; Review comment: This is easy enough to remove. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
14. jdeppe-pivotal commented on issue #1011: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1011#issuecomment-342205242 I'm going to close this and re-submit under my userid in order to be able to apply the review changes. ---------- This is an automated message from the Apache Git Service. To respond to the message,

please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

15. jdeppe-pivotal closed pull request #1011: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1011 This is a PR merged from a forked repository. As GitHub hides the original diff on merge, it is displayed below for the sake of provenance: As this is a foreign pull request (from a fork), the diff is supplied below (as it won't show otherwise due to GitHub magic): diff --git a/geode-assembly/build.gradle b/geode-assembly/build.gradle index 1f139a5ac9..146df75382 100755 --- a/geode-assembly/build.gradle +++ b/geode-assembly/build.gradle @@ -160,6 +160,7 @@ def cp = { // depedencies from geode-core it.contains('antlr') || it.contains('commons-io') || + it.contains('commons-beanutils') || it.contains('commons-collections') || it.contains('commons-lang') || it.contains('commons-logging') || @@ -168,6 +169,7 @@ def cp = { it.contains('commons-codec') || it.contains('fast-classpath-scanner') || it.contains('fastutil') || + it.contains('guava') || it.contains('jackson-annotations') || it.contains('jackson-core') || it.contains('jackson-databind') || @@ -193,19 +195,20 @@ def cp = { it.contains('log4j-slf4j-impl') || it.contains('shiro') || it.contains('slf4j-api') || + it.contains('spring-beans') || + it.contains('spring-context') || + it.contains('spring-context-support') || it.contains('spring-core') || it.contains('spring-shell') || it.contains('snappy') || it.contains('jgroups') || it.contains('netty') || - // dependencies from geode-lucene it.contains('lucene-analyzers-common') || it.contains('lucene-core') || it.contains('lucene-queries') || it.contains('lucene-queryparser') || it.contains('lucene-analyzers-phonetic') || - // dependencies from geode-protobuf it.contains('protobuf-java') } diff --git a/geode-assembly/src/test/java/org/apache/geode/management/internal/cli/commands/StatusLocatorRealGfshTest.java b/geode-assembly/src/test/java/org/apache/geode/management/internal/cli/commands/StatusLocatorRealGfshTest.java index a67dddcdd7..94ddc8f322 100644 --- a/geode-assembly/src/test/java/org/apache/geode/management/internal/cli/commands/StatusLocatorRealGfshTest.java +++ b/geode-assembly/src/test/java/org/apache/geode/management/internal/cli/commands/StatusLocatorRealGfshTest.java @@ -18,9 +18,9 @@ import org.junit.Test; import org.junit.experimental.categories.Category; +import org.apache.geode.test.junit.categories.AcceptanceTest; import org.apache.geode.test.junit.rules.gfsh.GfshRule; import org.apache.geode.test.junit.rules.gfsh.GfshScript; -import org.apache.geode.test.junit.categories.AcceptanceTest; @Category(AcceptanceTest.class) public class StatusLocatorRealGfshTest { @@ -30,7 +30,6 @@ @Test public void statusLocatorSucceedsWhenConnected() throws Exception { GfshScript.of("start locator --name=locator1").execute(gfshRule); - GfshScript.of("connect", "status locator --name=locator1").execute(gfshRule); } diff --git a/geode-assembly/src/test/resources/expected_jars.txt b/geode-assembly/src/test/resources/expected_jars.txt index bc49bcef62..7e3d438720 100644 --- a/geode-assembly/src/test/resources/expected_jars.txt +++ b/geode-assembly/src/test/resources/expected_jars.txt @@ -64,6 +64,7 @@ spring-aop spring-aspects spring-beans spring-context +spring-context-support spring-core spring-expression spring-hateoas diff --git a/geode-core/build.gradle b/geode-core/build.gradle index 3f47b0784e..263cc33138 100755 --- a/geode-core/build.gradle +++ b/geode-core/build.gradle @@ -52,6 +52,7 @@ dependencies { compile 'commons-io:commons-io:' + project.'commons-io.version' compile 'commons-validator:commons-validator:' + project.'commons-validator.version' compile 'commons-digester:commons-digester:' + project.'commons-digester.version' + compile 'com.google.guava:guava:' + project.'guava.version' compile 'commons-lang:commons-lang:' + project.'commons-lang.version' compile ('commons-modeler:commons-modeler:' + project.'commons-modeler.version') { @@ -111,9 +112,7 @@ dependencies { exclude module: 'aopalliance' exclude module: 'asm' exclude module: 'cglib' - exclude module: 'guava' exclude module: 'spring-aop' - exclude module: 'spring-context-support' ext.optional = true } compile ('org.iq80.snappy:snappy:' + project.'snappy-java.version') { diff --git a/geode-core/src/main/java/org/apache/geode/distributed/ConfigurationProperties.java b/geode-core/src/main/java/org/apache/geode/distributed/ConfigurationProperties.java index aae7d8a136..0e500a33ec 100644 --- a/geode-core/src/main/java/org/apache/geode/distributed/ConfigurationProperties.java +++ b/geode-core/src/main/java/org/apache/geode/distributed/ConfigurationProperties.java @@ -1860,7 +1860,10 @@ * <U>Default</U>: <code>""</code> * </p> * <U>Since</U>: GemFire 8.0 + * + * @deprecated Since Geode 1.4 (See GEODE-3007) */ + @Deprecated() String USER_COMMAND_PACKAGES = "user-command-packages"; /** * The static String definition of the "off-heap-memory-size" property <a diff --git a/geode-core/src/main/java/org/apache/geode/distributed/internal/AbstractDistributionConfig.java b/geode-core/src/main/java/org/apache/geode/distributed/internal/AbstractDistributionConfig.java index bfc5e1d5d4..3e5681b74c 100644 --- a/geode-core/src/main/java/org/apache/geode/distributed/internal/AbstractDistributionConfig.java +++ b/geode-core/src/main/java/org/apache/geode/distributed/internal/AbstractDistributionConfig.java @@ -1018,9 +1018,6 @@ public static Class _getAttributeType(String attName) { m.put(GROUPS, "A comma separated

list of all the groups this member belongs to." + " Defaults to \"\"."); - m.put(USER_COMMAND_PACKAGES, - "A comma separated list of the names of the packages containing classes that implement user commands."); - m.put(JMX_MANAGER, "If true then this member is willing to be a jmx manager. Defaults to false except on a locator."); m.put(JMX_MANAGER_START, diff --git a/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfig.java b/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfig.java index 958ea15028..bf353ed9ae 100644 --- a/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfig.java +++ b/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfig.java @@ -387,34 +387,6 @@ File DEFAULT_DEPLOY_WORKING_DIR = new File(System.getProperty("user.dir")); /** - * Returns the value of the {@link ConfigurationProperties#USER_COMMAND_PACKAGES} property - */ - @ConfigAttributeGetter(name = USER_COMMAND_PACKAGES) - String getUserCommandPackages(); - - /** - * Sets the system's user command path. - * - * @throws IllegalArgumentException if the specified value is not acceptable. - * @throws org.apache.geode.UnmodifiableException if this attribute can not be modified. - * @throws org.apache.geode.GemFireIOException if the set failure is caused by an error when - * writing to the system's configuration file. - */ - @ConfigAttributeSetter(name = USER_COMMAND_PACKAGES) - void setUserCommandPackages(String value); - - /** - * The name of the {@link ConfigurationProperties#USER_COMMAND_PACKAGES} property. - */ - @ConfigAttribute(type = String.class) - String USER_COMMAND_PACKAGES_NAME = USER_COMMAND_PACKAGES; - - /** - * The default value of the {@link ConfigurationProperties#USER_COMMAND_PACKAGES} property - */ - String DEFAULT_USER_COMMAND_PACKAGES = ""; - /** * Returns the value of the {@link ConfigurationProperties#LOG_FILE} property * * @return <code>null</code> if logging information goes to standard out diff --git a/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfigImpl.java b/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfigImpl.java index abaa83c822..3862b22c20 100644 --- a/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfigImpl.java +++ b/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfigImpl.java @@ -625,8 +625,6 @@ private Map<String, ConfigSource> sourceMap = Collections.synchronizedMap(new HashMap<String, ConfigSource>()); - protected String userCommandPackages = DEFAULT_USER_COMMAND_PACKAGES; - /** * "off-heap-memory-size" with value of "" or "<size> [g|m]" */ @@ -759,7 +757,6 @@ public DistributionConfigImpl(DistributionConfig other) { this.redisPort = other.getRedisPort(); this.redisBindAddress = other.getRedisBindAddress(); this.redisPassword = other.getRedisPassword(); - this.userCommandPackages = other.getUserCommandPackages(); // following added for 8.0 this.enableSharedConfiguration = other.getEnableClusterConfiguration(); @@ -1834,10 +1831,6 @@ public int getAsyncMaxQueueSize() { return this.asyncMaxQueueSize; } - public String getUserCommandPackages() { - return this.userCommandPackages; - } - public int getHttpServicePort() { return this.httpServicePort; } @@ -1862,10 +1855,6 @@ public void setStartDevRestApi(boolean value) { this.startDevRestApi = value; } - public void setUserCommandPackages(String value) { - this.userCommandPackages = value; - } - public boolean getDeltaPropagation() { return this.deltaPropagation; } @@ -3013,9 +3002,8 @@ public boolean equals(final Object o) { .append(sslTrustStore, that.sslTrustStore) .append(sslTrustStorePassword, that.sslTrustStorePassword) .append(locatorSSLAlias, that.locatorSSLAlias).append(sslDefaultAlias, that.sslDefaultAlias) - .append(sourceMap, that.sourceMap).append(userCommandPackages, that.userCommandPackages) - .append(offHeapMemorySize, that.offHeapMemorySize).append(shiroInit, that.shiroInit) - .isEquals(); + .append(sourceMap, that.sourceMap).append(offHeapMemorySize, that.offHeapMemorySize) + .append(shiroInit, that.shiroInit).isEquals(); } /** @@ -3083,9 +3071,8 @@ public int hashCode() { .append(sslTrustStorePassword).append(sslWebServiceRequireAuthentication) .append(locatorSSLAlias).append(sslDefaultAlias).append(sourceMap) - .append(userCommandPackages).append(offHeapMemorySize).append(lockMemory).append(shiroInit) - .append(modifiable).toHashCode(); + .append(locatorSSLAlias).append(sslDefaultAlias).append(sourceMap).append(offHeapMemorySize) + .append(lockMemory).append(shiroInit).append(modifiable).toHashCode(); } /** diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/beans/MemberMBeanBridge.java b/geode-core/src/main/java/org/apache/geode/management/internal/beans/MemberMBeanBridge.java index 5b911d5636..1d65ec949e 100644 --- a/geode-core/src/main/java/org/apache/geode/management/internal/beans/MemberMBeanBridge.java +++ b/geode-core/src/main/java/org/apache/geode/management/internal/beans/MemberMBeanBridge.java @@ -338,8

+338,7 @@ public MemberMBeanBridge(InternalCache cache, SystemManagementService service) { this.config = system.getConfig(); try { - this.commandProcessor = - new OnlineCommandProcessor(system.getProperties(), cache.getSecurityService()); + this.commandProcessor = new OnlineCommandProcessor(cache.getSecurityService()); } catch (Exception e) { commandServiceInitError = e.getMessage(); logger.info(LogMarker.CONFIG, "Command processor could not be initialized. {}", diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/CommandManager.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/CommandManager.java index 01605986c0..6c23201e94 100644 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/CommandManager.java +++ b/geode-core/src/main/java/org/apache/geode/management/internal/cli/CommandManager.java @@ -14,293 +14,96 @@ */ package org.apache.geode.management.internal.cli; -import static org.apache.geode.distributed.ConfigurationProperties.USER_COMMAND_PACKAGES; -import java.lang.reflect.Method; -import java.util.ArrayList; -import java.util.HashSet; -import java.util.Iterator; -import java.util.List; -import java.util.Properties; -import java.util.ServiceConfigurationError; -import java.util.ServiceLoader; +import static java.util.stream.Collectors.toSet; + +import java.util.Objects; import java.util.Set; -import java.util.StringTokenizer; +import java.util.function.Predicate; +import com.google.common.collect.ImmutableSet; +import org.springframework.shell.commands.ConsoleCommands; +import org.springframework.shell.commands.ExitCommands; +import org.springframework.shell.commands.HelpCommands; import org.springframework.shell.converters.EnumConverter; import org.springframework.shell.converters.SimpleFileConverter; import org.springframework.shell.core.CommandMarker; import org.springframework.shell.core.Converter; -import org.springframework.shell.core.MethodTarget; -import org.springframework.shell.core.annotation.CliAvailabilityIndicator; -import org.springframework.shell.core.annotation.CliCommand; -import org.apache.geode.distributed.ConfigurationProperties; -import org.apache.geode.distributed.internal.DistributionConfig; import org.apache.geode.internal.ClassPathLoader; -import org.apache.geode.management.internal.cli.commands.GfshCommand; import org.apache.geode.management.internal.cli.help.Helper; -import org.apache.geode.management.internal.cli.shell.Gfsh; -import org.apache.geode.management.internal.cli.util.ClasspathScanLoadHelper; -/** - * this only takes care of loading all available command markers and converters from the application - * - * @since GemFire 7.0 - */ public class CommandManager { - public static final String USER_CMD_PACKAGES_PROPERTY = - DistributionConfig.GEMFIRE_PREFIX + USER_COMMAND_PACKAGES; - public static final String USER_CMD_PACKAGES_ENV_VARIABLE = "GEMFIRE_USER_COMMAND_PACKAGES"; - private static final Object INSTANCE_LOCK = new Object(); - - private final Helper helper = new Helper(); - - private final List<Converter<?>> converters = new ArrayList<Converter<?>>(); - private final List<CommandMarker> commandMarkers = new ArrayList<>(); + // Skip some of the Converters from Spring Shell for our customization + private static final Set<Class> EXCLUDED_CLASSES = ImmutableSet.of(SimpleFileConverter.class, + EnumConverter.class, ExitCommands.class, HelpCommands.class, ConsoleCommands.class); - private Properties cacheProperties; - private LogWrapper logWrapper; + private final LogWrapper logWrapper = LogWrapper.getInstance(); + private final Set<CommandMarker> commandMarkers; + private final Set<Converter> converters; + private final Helper helper; - /** - * this constructor is used from Gfsh VM. We are getting the user-command-package from system - * environment. used by Gfsh. - */ public CommandManager() { - this(null); + helper = new Helper(); + converters = loadConverters(); + commandMarkers = loadCommandMarkers(); } - /** - * this is used when getting the instance in a cache server. We are getting the user-command-package from distribution properties. used by OnlineCommandProcessor. - */ - public CommandManager(final Properties cacheProperties) { - if (cacheProperties != null) { - this.cacheProperties = cacheProperties; - } - logWrapper = LogWrapper.getInstance(); - loadCommands(); + private Set<Converter> loadConverters() { + Set<Converter> converters = instantiateAllClassesImplementing(Converter.class); + raiseExceptionIfEmpty(converters, "converters"); + converters.forEach(this::setContextIfCommandManagerAware); + return converters; } - private void loadUserCommands() { - final Set<String> userCommandPackages = new HashSet<String>(); - - // Find by packages specified by the system property - if (System.getProperty(USER_CMD_PACKAGES_PROPERTY) != null) { - StringTokenizer tokenizer = - new StringTokenizer(System.getProperty(USER_CMD_PACKAGES_PROPERTY), ","); - while

(tokenizer.hasMoreTokens()) { - userCommandPackages.add(tokenizer.nextToken()); - } - } - - // Find by packages specified by the environment variable - if (System.getenv().containsKey(USER_CMD_PACKAGES_ENV_VARIABLE)) { - StringTokenizer tokenizer = - new StringTokenizer(System.getenv().get(USER_CMD_PACKAGES_ENV_VARIABLE), ","); - while (tokenizer.hasMoreTokens()) { - userCommandPackages.add(tokenizer.nextToken()); - } - } - - // Find by packages specified in the distribution config - if (this.cacheProperties != null) { - String cacheUserCmdPackages = - this.cacheProperties.getProperty(ConfigurationProperties.USER_COMMAND_PACKAGES); - if (cacheUserCmdPackages != null && !cacheUserCmdPackages.isEmpty()) { - StringTokenizer tokenizer = new StringTokenizer(cacheUserCmdPackages, ","); - while (tokenizer.hasMoreTokens()) { - userCommandPackages.add(tokenizer.nextToken()); - } - } - } + private Set<CommandMarker> loadCommandMarkers() { + Set<CommandMarker> commandMarkers = instantiateAllClassesImplementing(CommandMarker.class); + raiseExceptionIfEmpty(commandMarkers, "commandMarkers"); - // Load commands found in all of the packages - for (String userCommandPackage : userCommandPackages) { - try { - Set<Class<?>> foundClasses = ClasspathScanLoadHelper - .scanPackageForClassesImplementing(userCommandPackage, CommandMarker.class); - for (Class<?> klass : foundClasses) { - try { - add((CommandMarker) klass.newInstance()); - } catch (Exception e) { - logWrapper.warning("Could not load User Commands from: " + klass + " due to " - + e.getLocalizedMessage()); // continue - } - } - raiseExceptionIfEmpty(foundClasses, "User Command"); - } catch (IllegalStateException e) { - logWrapper.warning(e.getMessage(), e); - throw e; - } - } - - /** - * Loads commands via {@link ServiceLoader} from {@link ClassPathLoader}. - * - * @since GemFire 8.1 - */ - private void loadPluginCommands() { - final Iterator<CommandMarker> iterator = ServiceLoader - .load(CommandMarker.class, ClassPathLoader.getLatest().asClassLoader()).iterator(); - while (iterator.hasNext()) { - try { - final CommandMarker commandMarker = iterator.next(); - try { - add(commandMarker); - } catch (Exception e) { - logWrapper.warning("Could not load Command from: " + commandMarker.getClass() + " due to " - + e.getLocalizedMessage(), e); // continue - } - } catch (ServiceConfigurationError e) { - logWrapper.severe("Could not load Command: " + e.getLocalizedMessage(), e); // continue - } - } + commandMarkers.forEach(this::setContextIfCommandManagerAware); + commandMarkers.forEach(helper::registerCommand); + return commandMarkers; } + private <T> Set<T> instantiateAllClassesImplementing(Class<T> implementedInterface) { + Set<Class<? extends T>> classes = + ClasspathScanLoadHelper.scanClasspathForClassesImplementing(implementedInterface); - private void loadCommands() { - loadUserCommands(); + Predicate<Class<? extends T>> classIsNotExcluded = aClass -> !EXCLUDED_CLASSES.contains(aClass); - loadPluginCommands(); - loadGeodeCommands(); - loadConverters(); + return classes.stream().filter(classIsNotExcluded).map(this::instantiateClass) + .filter(Objects::nonNull).collect(toSet()); } - private void loadConverters() { - Set<Class<?>> foundClasses; - // Converters + private <T> T instantiateClass(Class<T> classToInstantiate) { try { - foundClasses = ClasspathScanLoadHelper.scanPackageForClassesImplementing(- "org.apache.geode.management.internal.cli.converters", Converter.class); - for (Class<?> klass : foundClasses) { - try { - Converter<?> object = (Converter<?>) klass.newInstance(); - add(object); - } catch (Exception e) { - logWrapper.warning(- "Could not load Converter from: " + klass + " due to " + e.getLocalizedMessage()); // continue - } - } - raiseExceptionIfEmpty(foundClasses, "Converters"); - - // Spring shell's converters - foundClasses = ClasspathScanLoadHelper.scanPackageForClassesImplementing(- "org.springframework.shell.converters", Converter.class); - for (Class<?> klass : foundClasses) { - if (!SHL_CONVERTERS_TOSKIP.contains(klass)) { - try { - add((Converter<?>) klass.newInstance()); - } catch (Exception e) { - logWrapper.warning(- "Could not load Converter from: " + klass + " due to " + e.getLocalizedMessage()); // continue - } - } - } - raiseExceptionIfEmpty(foundClasses, "Basic Converters"); - } catch (IllegalStateException e) { - logWrapper.warning(e.getMessage(), e); - throw e; + return classToInstantiate.newInstance(); + } catch (Exception e) { + logWrapper.warning("Could not load command or converter from: " + classToInstantiate, e); } + return null; } - private void loadGeodeCommands() { - // CommandMarkers - Set<Class<?>> foundClasses; - try { - // geode's commands - foundClasses = ClasspathScanLoadHelper.scanPackageForClassesImplementing(- GfshCommand.class.getPackage().getName(), CommandMarker.class); - - for (Class<?> klass : foundClasses) { - try { - add((CommandMarker) klass.newInstance()); - } catch (Exception e) { - logWrapper.warning(- "Could not load Command from: " + klass + " due to " + e.getLocalizedMessage()); // continue - } - } - raiseExceptionIfEmpty(foundClasses, "Commands"); - - // do not add Spring shell's commands for now. When we add it, we need to tell the parser that - // these are offline commands. - } catch (IllegalStateException e) { - logWrapper.warning(e.getMessage(), e); - throw e; + private void setContextIfCommandManagerAware(Object commandOrConverter) { + if (CommandManagerAware.class.isAssignableFrom(commandOrConverter.getClass())) { + ((CommandManagerAware) commandOrConverter).setCommandManager(this); } } - /** Skip some of the

Converters from Spring Shell for our customization */ - private static List<Class> SHL_CONVERTERS_TOSKIP = new ArrayList(); - static { - // skip springs SimpleFileConverter to use our own FilePathConverter - SHL_CONVERTERS_TOSKIP.add(SimpleFileConverter.class); - // skip spring's EnumConverter to use our own EnumConverter - SHL_CONVERTERS_TOSKIP.add(EnumConverter.class); - } - - public List<Converter<?>> getConverters() { - return converters; - } - - public List<CommandMarker> getCommandMarkers() { - return commandMarkers; - } - /** - * Method to add new Converter - * - * @param converter - */ - void add(Converter<?> converter) { - if (CommandManagerAware.class.isAssignableFrom(converter.getClass())) { - ((CommandManagerAware) converter).setCommandManager(this); - } - converters.add(converter); - } - /** - * Method to add new Commands to the parser - * - * @param commandMarker - */ - void add(CommandMarker commandMarker) { - if (CommandManagerAware.class.isAssignableFrom(commandMarker.getClass())) { - ((CommandManagerAware) commandMarker).setCommandManager(this); - } - commandMarkers.add(commandMarker); - for (Method method : commandMarker.getClass().getMethods()) { - CliCommand cliCommand = method.getAnnotation(CliCommand.class); - CliAvailabilityIndicator availability = method.getAnnotation(CliAvailabilityIndicator.class); - if (cliCommand == null && availability == null) { - continue; - } - - if (cliCommand != null) { - helper.addCommand(cliCommand, method); - } - - if (availability != null) { - helper.addAvailabilityIndicator(availability, new MethodTarget(method, commandMarker)); - } + private static void raiseExceptionIfEmpty(Set<?> foundClasses, String classType) + throws IllegalStateException { + if (foundClasses == null || foundClasses.isEmpty()) { + throw new IllegalStateException("No " + classType + " were loaded. Check logs for errors."); } } - public Helper getHelper() { - return helper; - } - public String obtainHelp(String buffer) { int terminalWidth = -1; Gfsh gfsh = Gfsh.getCurrentInstance(); @@ -314,4 +117,15 @@ public String obtainHint(String topic) { return helper.getHint(topic); } + public Set<Converter> getConverters() { + return converters; + } + + + public Set<CommandMarker> getCommandMarkers() { + return commandMarkers; + } + + + public Helper getHelper() { + return helper; + } } diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/GfshParser.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/GfshParser.java index 9cebc63c34..179d4ec44b 100755 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/GfshParser.java +++ b/geode-core/src/main/java/org/apache/geode/management/internal/cli/GfshParser.java @@ -57,10 +57,14 @@ public GfshParser(CommandManager commandManager) { add(command); } - for (Converter<?> converter : commandManager.getConverters()) { + for (Converter converter : commandManager.getConverters()) { if (converter.getClass().isAssignableFrom(ArrayConverter.class)) { ArrayConverter arrayConverter = (ArrayConverter) converter; - arrayConverter.setConverters(new HashSet<>(commandManager.getConverters())); + HashSet<Converter<?>> converters = new HashSet<>(); + for (Converter c : commandManager.getConverters()) { + converters.add(c); + } + arrayConverter.setConverters(converters); } add(converter); } diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/help/Helper.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/help/Helper.java index 352501324c..a712e37f62 100644 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/help/Helper.java +++ b/geode-core/src/main/java/org/apache/geode/management/internal/cli/help/Helper.java @@ -14,18 +14,10 @@ */ package org.apache.geode.management.internal.cli.help; -import org.apache.commons.lang.StringUtils; -import org.apache.geode.management.cli.CliMetaData; -import org.apache.geode.management.internal.cli.GfshParser; -import org.apache.geode.management.internal.cli.i18n.CliStrings; -import org.springframework.shell.core.MethodTarget; -import org.springframework.shell.core.annotation.CliAvailabilityIndicator; -import org.springframework.shell.core.annotation.CliCommand; -import org.springframework.shell.core.annotation.CliOption; +import static java.util.stream.Collectors.joining; import java.lang.annotation.Annotation; import java.lang.reflect.Method; -import java.util.ArrayList; import java.util.Arrays; import java.util.Collection; import java.util.Collections; @@ -35,6 +27,17 @@ import java.util.Set; import java.util.TreeMap; +import org.apache.commons.lang.StringUtils; +import org.springframework.shell.core.CommandMarker; +import org.springframework.shell.core.MethodTarget; +import org.springframework.shell.core.annotation.CliAvailabilityIndicator; +import org.springframework.shell.core.annotation.CliCommand; +import org.springframework.shell.core.annotation.CliOption; + + +import org.apache.geode.management.cli.CliMetaData; +import org.apache.geode.management.internal.cli.GfshParser; +import org.apache.geode.management.internal.cli.i18n.CliStrings; + /** * @@ -93,7 +96,25 @@ private void initTopic(String topic, String desc) { topics.put(topic, new Topic(topic, desc)); } - public void

addCommand(CliCommand command, Method commandMethod) { + public void registerCommand(CommandMarker commandMarker) { + for (Method method : commandMarker.getClass().getMethods()) { + CliCommand cliCommand = method.getAnnotation(CliCommand.class); + CliAvailabilityIndicator availability = method.getAnnotation(CliAvailabilityIndicator.class); + if (cliCommand == null && availability == null) { + continue; + } + + if (cliCommand != null) { + addCommand(cliCommand, method); + } + + if (availability != null) { + addAvailabilityIndicator(availability, new MethodTarget(method, commandMarker)); + } + } + } + + protected void addCommand(CliCommand command, Method commandMethod) { // put all the command synonyms in the command map Arrays.stream(command.value()).forEach(cmd -> { commands.put(cmd, commandMethod); @@ -119,7 +140,8 @@ public void addCommand(CliCommand command, Method commandMethod) { }); } - public void addAvailabilityIndicator(CliAvailabilityIndicator availability, MethodTarget target) { + private void addAvailabilityIndicator(CliAvailabilityIndicator availability, + MethodTarget target) { Arrays.stream(availability.value()).forEach(command -> { availabilityIndicators.put(command, target); }); @@ -151,9 +173,7 @@ public String getHint(String buffer) { builder.append(CliStrings.HINT__MSG__TOPICS_AVAILABLE).append(GfshParser.LINE_SEPARATOR) .append(GfshParser.LINE_SEPARATOR); - List<String> sortedTopics = new ArrayList<>(topics.keySet()); - Collections.sort(sortedTopics); - sortedTopics.stream() + topics.keySet().stream().sorted() .forEachOrdered(topic -> builder.append(topic).append(GfshParser.LINE_SEPARATOR)); return builder.toString(); } @@ -164,8 +184,7 @@ public String getHint(String buffer) { } builder.append(topic.desc).append(GfshParser.LINE_SEPARATOR).append(GfshParser.LINE_SEPARATOR); - Collections.sort(topic.relatedCommands); - topic.relatedCommands.stream().forEachOrdered(command -> builder.append(command.command) + topic.relatedCommands.stream().sorted().forEach(command -> builder.append(command.command) .append(": ").append(command.desc).append(GfshParser.LINE_SEPARATOR)); return builder.toString(); } @@ -174,7 +193,7 @@ public String getHint(String buffer) { return topics.keySet(); } - boolean isAvailable(String command) { + private boolean isAvailable(String command) { MethodTarget target = availabilityIndicators.get(command); if (target == null) { return true; @@ -260,16 +279,12 @@ HelpBlock getOptionDetail(CliOption cliOption) { HelpBlock optionNode = new HelpBlock(getPrimaryKey(cliOption)); String help = cliOption.help(); optionNode.addChild(new HelpBlock((StringUtils.isNotBlank(help) ? help : ""))); - if (getSynonyms(cliOption).size() > 0) { - StringBuilder builder = new StringBuilder(); - for (String string : getSynonyms(cliOption)) { - if (builder.length() > 0) { - builder.append(","); - } - builder.append(string); - } - optionNode.addChild(new HelpBlock(SYNONYMS_SUB_NAME + builder.toString())); + String synonyms = getSynonyms(cliOption).stream().collect(joining(",")); + if (StringUtils.isNotEmpty(synonyms)) { + optionNode.addChild(new HelpBlock(SYNONYMS_SUB_NAME + synonyms)); } optionNode.addChild(new HelpBlock(REQUIRED_SUB_NAME + ((cliOption.mandatory()) ? TRUE_TOKEN : FALSE_TOKEN))); if (!isNullOrBlank(cliOption.specifiedDefaultValue())) { @@ -358,18 +373,14 @@ private static String getPrimaryKey(CliOption option) { } private static List<String> getSynonyms(CliOption option) { - List<String> synonyms = new ArrayList<>(); String[] keys = option.key(); if (keys.length < 2) - return synonyms; + return Collections.emptyList(); // if the primary key is empty (like sh and help command), then there should be no synonyms. if ("".equals(keys[0])) - return synonyms; + return Collections.emptyList(); - for (int i = 1; i < keys.length; i++) { - synonyms.add(keys[i]); - } - return synonyms; + return Arrays.asList(keys).subList(1, keys.length); } private static boolean isNullOrBlank(String value) { diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/MemberCommandService.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/MemberCommandService.java index 6130117d2e..a4c6f59802 100644 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/MemberCommandService.java +++ b/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/MemberCommandService.java @@ -35,8 +35,7 @@ public MemberCommandService(InternalCache cache) throws CommandServiceException { this.cache = cache; try { - this.onlineCommandProcessor = new OnlineCommandProcessor(- cache.getDistributedSystem().getProperties(), cache.getSecurityService()); + this.onlineCommandProcessor = new OnlineCommandProcessor(cache.getSecurityService()); } catch (Exception e) { throw new CommandServiceException("Could not load commands.", e); } diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessor.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessor.java index 7fa4acba51..38ab364392 100644 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessor.java +++ b/geode-

core/src/main/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessor.java @@ -18,7 +18,6 @@ import java.lang.reflect.Method; import java.util.Collections; import java.util.Map; -import java.util.Properties; import org.springframework.shell.core.Parser; import org.springframework.shell.event.ParseResult; @@ -26,7 +25,6 @@ import org.apache.geode.annotations.TestingOnly; import org.apache.geode.internal.security.SecurityService; -import org.apache.geode.internal.security.SecurityServiceFactory; import org.apache.geode.management.cli.CliMetaData; import org.apache.geode.management.cli.CommandProcessingException; import org.apache.geode.management.cli.Result; @@ -49,20 +47,14 @@ private final SecurityService securityService; - @TestingOnly - public OnlineCommandProcessor() throws ClassNotFoundException, IOException { - this(new Properties(), SecurityServiceFactory.create()); - } - - public OnlineCommandProcessor(Properties cacheProperties, SecurityService securityService) + public OnlineCommandProcessor(SecurityService securityService) throws ClassNotFoundException, IOException { - this(cacheProperties, securityService, new CommandExecutor()); + this(securityService, new CommandExecutor()); } @TestingOnly - public OnlineCommandProcessor(Properties cacheProperties, SecurityService securityService, - CommandExecutor commandExecutor) { - this.gfshParser = new GfshParser(new CommandManager(cacheProperties)); + public OnlineCommandProcessor(SecurityService securityService, CommandExecutor commandExecutor) { + this.gfshParser = new GfshParser(new CommandManager()); this.executor = commandExecutor; this.securityService = securityService; } diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/util/ClasspathScanLoadHelper.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/util/ClasspathScanLoadHelper.java index d3bc7d3094..3713d5a9ab 100644 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/util/ClasspathScanLoadHelper.java +++ b/geode-core/src/main/java/org/apache/geode/management/internal/cli/util/ClasspathScanLoadHelper.java @@ -16,30 +16,39 @@ import static java.util.stream.Collectors.toSet; -import io.github.lukehutch.fastclasspathscanner.FastClasspathScanner; - import java.lang.reflect.Modifier; import java.util.HashSet; import java.util.Set; +import io.github.lukehutch.fastclasspathscanner.FastClasspathScanner; +import io.github.lukehutch.fastclasspathscanner.matchprocessor.ImplementingClassMatchProcessor; + /** * Utility class to scan class-path & load classes. * * @since GemFire 7.0 */ public class ClasspathScanLoadHelper { - public static Set<Class<?>> scanPackageForClassesImplementing(String packageToScan, - Class<?> implementedInterface) { - Set<Class<?>> classesImplementing = new HashSet<>(); + public static <T> Set<Class<? extends T>> scanClasspathForClassesImplementing(+ Class<T> implementedInterface) { + return scanPackageForClassesImplementing("", implementedInterface); + } + + public static <T> Set<Class<? extends T>> scanPackageForClassesImplementing(String packageToScan, + Class<T> implementedInterface) { + Set<Class<? extends T>> classesImplementing = new HashSet<>(); + ImplementingClassMatchProcessor<T> matchProcessor = classesImplementing::add; new FastClasspathScanner(packageToScan) - .matchClassesImplementing(implementedInterface, classesImplementing::add).scan(); + .matchClassesImplementing(implementedInterface, matchProcessor).scan(); return classesImplementing.stream().filter(ClasspathScanLoadHelper::isInstantiable) .collect(toSet()); } - private static boolean isInstantiable(Class<?> klass) { - int modifiers = klass.getModifiers(); + private static <T> boolean isInstantiable(Class<T> classToInstantiate) { + int modifiers = classToInstantiate.getModifiers(); return !Modifier.isAbstract(modifiers) && !Modifier.isInterface(modifiers) && Modifier.isPublic(modifiers); diff --git a/geode-core/src/test/java/org/apache/geode/distributed/internal/DistributionConfigJUnitTest.java b/geode-core/src/test/java/org/apache/geode/distributed/internal/DistributionConfigJUnitTest.java index 7e5f83f4f2..e874898d82 100644 --- a/geode-core/src/test/java/org/apache/geode/distributed/internal/DistributionConfigJUnitTest.java +++ b/geode-core/src/test/java/org/apache/geode/distributed/internal/DistributionConfigJUnitTest.java @@ -99,7 +99,7 @@ public void before() { @Test public void testGetAttributeNames() { String[] attNames = AbstractDistributionConfig._getAttNames(); - assertEquals(attNames.length, 157); + assertEquals(156, attNames.length); List boolList = new ArrayList(); List intList = new ArrayList(); @@ -135,7 +135,7 @@ public void testGetAttributeNames() { // are. assertEquals(29, boolList.size()); assertEquals(33, intList.size()); - assertEquals(86, stringList.size()); + assertEquals(85, stringList.size()); assertEquals(5, fileList.size()); assertEquals(4, otherList.size()); } diff --git a/geode-core/src/test/java/org/apache/geode/management/internal/cli/ClasspathScanLoadHelperJUnitTest.java b/geode-core/src/test/java/org/apache/geode/management/internal/cli/ClasspathScanLoadHelperJUnitTest.java index 827eb9c400..04794d4559 100644 --- a/geode-core/src/test/java/org/apache/geode/management/internal/cli/ClasspathScanLoadHelperJUnitTest.java +++ b/geode-

core/src/test/java/org/apache/geode/management/internal/cli/ClasspathScanLoadHelperJUnitTest.java @@ -14,11 +14,11 @@ */ package org.apache.geode.management.internal.cli; -import static org.junit.Assert.*; +import static org.junit.Assert.assertEquals; +import static org.junit.Assert.assertTrue; import java.util.Set; -import org.apache.geode.management.internal.cli.util.ClasspathScanLoadHelper; import org.junit.Test; import org.junit.experimental.categories.Category; @@ -28,6 +28,7 @@ import org.apache.geode.management.internal.cli.domain.Impl12; import org.apache.geode.management.internal.cli.domain.Interface1; import org.apache.geode.management.internal.cli.domain.Interface2; +import org.apache.geode.management.internal.cli.util.ClasspathScanLoadHelper; import org.apache.geode.test.junit.categories.UnitTest; @Category(UnitTest.class) @@ -35,12 +36,12 @@ private final String PACKAGE_NAME = "org.apache.geode.management.internal.cli.domain"; private final String WRONG_PACKAGE_NAME = "org.apache.geode.management.internal.cli.domain1"; - private final Class<?> INTERFACE1 = Interface1.class; - private final Class<?> NO_IMPL_INTERFACE = Versionable.class; - private final Class<?> INTERFACE2 = Interface2.class; - private final Class<?> IMPL1 = Impl1.class; - private final Class<?> IMPL2 = Impl12.class; - private final Class<?> ABSTRACT_IMPL = AbstractImpl.class; + private final Class INTERFACE1 = Interface1.class; + private final Class NO_IMPL_INTERFACE = Versionable.class; + private final Class INTERFACE2 = Interface2.class; + private final Class IMPL1 = Impl1.class; + private final Class IMPL2 = Impl12.class; + private final Class ABSTRACT_IMPL = AbstractImpl.class; @Test public void testLoadAndGet() throws Exception { diff --git a/geode-core/src/test/java/org/apache/geode/management/internal/cli/CommandManagerJUnitTest.java b/geode-core/src/test/java/org/apache/geode/management/internal/cli/CommandManagerJUnitTest.java index cde0b236a7..9c3ebf57a8 100644 --- a/geode-core/src/test/java/org/apache/geode/management/internal/cli/CommandManagerJUnitTest.java +++ b/geode-core/src/test/java/org/apache/geode/management/internal/cli/CommandManagerJUnitTest.java @@ -18,20 +18,12 @@ import static org.junit.Assert.assertNotNull; import static org.junit.Assert.assertTrue; -import java.util.List; - import org.junit.Before; import org.junit.Test; import org.junit.experimental.categories.Category; import org.springframework.shell.core.CommandMarker; -import org.springframework.shell.core.Completion; -import org.springframework.shell.core.Converter; -import org.springframework.shell.core.MethodTarget; -import org.springframework.shell.core.annotation.CliAvailabilityIndicator; import org.springframework.shell.core.annotation.CliCommand; -import org.springframework.shell.core.annotation.CliOption; -import org.apache.geode.management.cli.CliMetaData; import org.apache.geode.management.cli.Result; import org.apache.geode.management.internal.security.ResourceOperation; import org.apache.geode.security.ResourcePermission.Operation; @@ -43,49 +35,6 @@ */ @Category(UnitTest.class) public class CommandManagerJUnitTest { - - private static final String COMMAND1_NAME = "command1"; - private static final String COMMAND1_NAME_ALIAS = "command1_alias"; - private static final String COMMAND2_NAME = "c2"; - - private static final String COMMAND1_HELP = "help for " + COMMAND1_NAME; - // ARGUMENTS - private static final String ARGUMENT1_NAME = "argument1"; - private static final String ARGUMENT1_HELP = "help for argument1"; - private static final String ARGUMENT1_CONTEXT = "context for argument 1"; - private static final Completion[] ARGUMENT1_COMPLETIONS = - {new Completion("arg1"), new Completion("arg1alt")}; - private static final String ARGUMENT2_NAME = "argument2"; - private static final String ARGUMENT2_CONTEXT = "context for argument 2"; - private static final String ARGUMENT2_HELP = "help for argument2"; - private static final String ARGUMENT2_UNSPECIFIED_DEFAULT_VALUE = - "{unspecified default value for argument2}"; - private static final Completion[] ARGUMENT2_COMPLETIONS = - {new Completion("arg2"), new Completion("arg2alt")}; - - // OPTIONS - private static final String OPTION1_NAME = "option1"; - private static final String OPTION1_SYNONYM = "opt1"; - private static final String OPTION1_HELP = "help for option1"; - private static final String OPTION1_CONTEXT = "context for option1"; - private static final String OPTION1_SPECIFIED_DEFAULT_VALUE = - "{specified default value for option1}"; - private static final Completion[] OPTION1_COMPLETIONS = - {new Completion("option1"), new Completion("option1Alternate")}; - private static final String OPTION2_NAME = "option2"; - private static final String OPTION2_HELP = "help for option2"; - private static final String OPTION2_CONTEXT = "context for option2"; - private static final String OPTION2_SPECIFIED_DEFAULT_VALUE = - "{specified default value for option2}"; - private static final String OPTION3_NAME = "option3"; - private static final String OPTION3_SYNONYM = "opt3"; - private static final String OPTION3_HELP = "help for option3"; - private static final String OPTION3_CONTEXT = "context for option3"; - private static final String OPTION3_SPECIFIED_DEFAULT_VALUE = - "{specified default value for option3}"; - private static final String OPTION3_UNSPECIFIED_DEFAULT_VALUE = - "{unspecified default value for option3}"; - private CommandManager commandManager; @Before @@ -111,11 +60,7 @@ public void

testCommandManagerInstance() throws Exception { assertNotNull(commandManager); } - /** - * Tests {@link CommandManager#loadPluginCommands()}. - * - * @since GemFire 8.1 - */ + @Test public void testCommandManagerLoadPluginCommands() throws Exception { assertNotNull(commandManager); @@ -123,97 +68,10 @@ public void testCommandManagerLoadPluginCommands() throws Exception { // see META-INF/services/org.springframework.shell.core.CommandMarker service loader file. assertTrue("Should find listed plugin.", commandManager.getHelper().getCommands().contains("mock plugin command")); - assertTrue("Should not find unlisted plugin.", - !commandManager.getHelper().getCommands().contains("mock plugin command unlisted")); + assertThat(+ commandManager.getCommandMarkers().stream().anyMatch(c -> c instanceof MockPluginCommand)); } - /** - * class that represents dummy commands - */ - public static class Commands implements CommandMarker { - - @CliCommand(value = {COMMAND1_NAME, COMMAND1_NAME_ALIAS}, help = COMMAND1_HELP) - @CliMetaData(shellOnly = true, relatedTopic = {"relatedTopicOfCommand1"}) - @ResourceOperation(resource = Resource.CLUSTER, operation = Operation.READ) - public static String command1(- @CliOption(key = ARGUMENT1_NAME, optionContext = ARGUMENT1_CONTEXT, help = ARGUMENT1_HELP, - mandatory = true) String argument1, - @CliOption(key = ARGUMENT2_NAME, optionContext = ARGUMENT2_CONTEXT, help = ARGUMENT2_HELP, - unspecifiedDefaultValue = ARGUMENT2_UNSPECIFIED_DEFAULT_VALUE) String argument2, - @CliOption(key = {OPTION1_NAME, OPTION1_SYNONYM}, help = OPTION1_HELP, mandatory = true, - optionContext = OPTION1_CONTEXT, - specifiedDefaultValue = OPTION1_SPECIFIED_DEFAULT_VALUE) String option1, - @CliOption(key = {OPTION2_NAME}, help = OPTION2_HELP, optionContext = OPTION2_CONTEXT, - specifiedDefaultValue = OPTION2_SPECIFIED_DEFAULT_VALUE) String option2, - @CliOption(key = {OPTION3_NAME, OPTION3_SYNONYM}, help = OPTION3_HELP, - optionContext = OPTION3_CONTEXT, - unspecifiedDefaultValue = OPTION3_UNSPECIFIED_DEFAULT_VALUE, - specifiedDefaultValue = OPTION3_SPECIFIED_DEFAULT_VALUE) String option3) { - return null; - } - - @CliCommand(value = {COMMAND2_NAME}) - @ResourceOperation(resource = Resource.CLUSTER, operation = Operation.READ) - public static String command2() { - return null; - } - - @CliCommand(value = {"testParamConcat"}) - @ResourceOperation(resource = Resource.CLUSTER, operation = Operation.READ) - public static Result testParamConcat(@CliOption(key = {"string"}) String string, - @CliOption(key = {"stringArray"}) String[] stringArray, - @CliOption(key = {"integer"}) Integer integer, - @CliOption(key = {"colonArray"}) String[] colonArray) { - return null; - } - - @CliCommand(value = {"testMultiWordArg"}) - @ResourceOperation(resource = Resource.CLUSTER, operation = Operation.READ) - public static Result testMultiWordArg(@CliOption(key = "arg1") String arg1, - @CliOption(key = "arg2") String arg2) { - return null; - } - - @CliAvailabilityIndicator({COMMAND1_NAME}) - public boolean isAvailable() { - return true; // always available on server - } - } - - /** - * Used by testCommandManagerLoadPluginCommands - */ - private static class SimpleConverter implements Converter<String> { - - @Override - public boolean supports(Class<?> type, String optionContext) { - return type.isAssignableFrom(String.class); - } - - @Override - public String convertFromText(String value, Class<?> targetType, String optionContext) { - return value; - } - - @Override - public boolean getAllPossibleValues(List<Completion> completions, Class<?> targetType, - String existingData, String context, MethodTarget target) { - if (context.equals(ARGUMENT1_CONTEXT)) { - for (Completion completion : ARGUMENT1_COMPLETIONS) { - completions.add(completion); - } - } else if (context.equals(ARGUMENT2_CONTEXT)) { - for (Completion completion : ARGUMENT2_COMPLETIONS) { - completions.add(completion); - } - } else if (context.equals(OPTION1_CONTEXT)) { - for (Completion completion : OPTION1_COMPLETIONS) { - completions.add(completion); - } - } - return true; - } - } public static class MockPluginCommand implements CommandMarker { @CliCommand(value = "mock plugin command") @@ -222,13 +80,4 @@ public Result mockPluginCommand() { return null; } } - - public static class MockPluginCommandUnlisted implements CommandMarker { - @CliCommand(value = "mock plugin command unlisted") - @ResourceOperation(resource = Resource.CLUSTER, operation = Operation.READ) - public Result mockPluginCommandUnlisted() { - return null; - } - } - } diff --git a/geode-core/src/test/java/org/apache/geode/management/internal/cli/commands/CliCommandTestBase.java b/geode-core/src/test/java/org/apache/geode/management/internal/cli/commands/CliCommandTestBase.java index 20314faa9f..d79993201b 100644 --- a/geode-core/src/test/java/org/apache/geode/management/internal/cli/commands/CliCommandTestBase.java +++ b/geode-core/src/test/java/org/apache/geode/management/internal/cli/commands/CliCommandTestBase.java @@ -23,6 +23,19 @@ import static org.apache.geode.test.dunit.LogWriterUtils.getLogWriter; import static org.assertj.core.api.Assertions.assertThat; +import java.io.IOException; +import java.io.PrintStream; +import java.net.InetAddress; +import java.net.UnknownHostException; +import java.util.Map; +import java.util.Properties; +import java.util.Set; +import java.util.regex.Matcher; +import java.util.regex.Pattern; + +import org.junit.Rule; +import org.springframework.shell.core.CommandMarker; + import

org.apache.geode.cache.Cache; import org.apache.geode.internal.AvailablePortHelper; import org.apache.geode.management.ManagementService; @@ -37,18 +50,6 @@ import org.apache.geode.test.dunit.cache.internal.JUnit4CacheTestCase; import org.apache.geode.test.dunit.rules.DistributedRestoreSystemProperties; import org.apache.geode.test.junit.rules.serializable.SerializableTemporaryFolder; -import org.junit.Rule; -import org.springframework.shell.core.CommandMarker; - -import java.io.IOException; -import java.io.PrintStream; -import java.net.InetAddress; -import java.net.UnknownHostException; -import java.util.List; -import java.util.Map; -import java.util.Properties; -import java.util.regex.Matcher; -import java.util.regex.Pattern; /** * Base class for all the CLI/gfsh command dunit tests. @@ -79,7 +80,7 @@ public static boolean checkIfCommandsAreLoadedOrNot() { CommandManager manager = new CommandManager(); - List<CommandMarker> commands = manager.getCommandMarkers(); + Set<CommandMarker> commands = manager.getCommandMarkers(); return commands.size() >= 1; } diff --git a/geode-core/src/test/java/org/apache/geode/management/internal/cli/help/HelperUnitTest.java b/geode-core/src/test/java/org/apache/geode/management/internal/cli/help/HelperUnitTest.java index 58d09f09eb..5786e8a4c9 100644 --- a/geode-core/src/test/java/org/apache/geode/management/internal/cli/help/HelperUnitTest.java +++ b/geode-core/src/test/java/org/apache/geode/management/internal/cli/help/HelperUnitTest.java @@ -15,12 +15,14 @@ package org.apache.geode.management.internal.cli.help; +import static org.apache.geode.management.internal.cli.GfshParser.LINE_SEPARATOR; import static org.assertj.core.api.Assertions.assertThat; import static org.mockito.Mockito.mock; import static org.mockito.Mockito.when; -import static org.apache.geode.management.internal.cli.GfshParser.LINE_SEPARATOR; import org.apache.geode.test.junit.categories.UnitTest; +import java.lang.annotation.Annotation; +import java.lang.reflect.Method; import org.junit.Before; import org.junit.Test; import org.junit.experimental.categories.Category; @@ -29,8 +31,7 @@ import org.springframework.shell.core.annotation.CliCommand; import org.springframework.shell.core.annotation.CliOption; -import java.lang.annotation.Annotation; -import java.lang.reflect.Method; +import org.apache.geode.test.junit.categories.UnitTest; @Category(UnitTest.class) public class HelperUnitTest { diff --git a/geode-core/src/test/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessorTest.java b/geode-core/src/test/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessorTest.java index df00cf17aa..db7176ba4f 100644 --- a/geode-core/src/test/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessorTest.java +++ b/geode-core/src/test/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessorTest.java @@ -21,8 +21,6 @@ import static org.mockito.Mockito.mock; import static org.mockito.Mockito.when; -import java.util.Properties; - import org.junit.Before; import org.junit.Test; import org.junit.experimental.categories.Category; @@ -36,7 +34,6 @@ @Category(UnitTest.class) public class OnlineCommandProcessorTest { - Properties properties; SecurityService securityService; CommandExecutor executor; OnlineCommandProcessor onlineCommandProcessor; @@ -44,13 +41,12 @@ @Before public void before() { - properties = new Properties(); securityService = mock(SecurityService.class); executor = mock(CommandExecutor.class); result = mock(Result.class); when(executor.execute(any())).thenReturn(result); - onlineCommandProcessor = new OnlineCommandProcessor(properties, securityService, executor); + onlineCommandProcessor = new OnlineCommandProcessor(securityService, executor); } diff --git a/geode-core/src/test/resources/META-INF/services/org.springframework.shell.core.CommandMarker b/geode-core/src/test/resources/META-INF/services/org.springframework.shell.core.CommandMarker deleted file mode 100644 index 1bb7c6d141..0000000000 --- a/geode-core/src/test/resources/META-INF/services/org.springframework.shell.core.CommandMarker +++ /dev/null @@ -1,8 +0,0 @@ -# Mock command for org.apache.geode.management.internal.cli.CommandManagerJUnitTest.testLoadPluginCommands -org.apache.geode.management.internal.cli.CommandManagerJUnitTest$MockPluginCommand -# Should cause ServiceConfigurationException while iterating. -#TODO jbarrett - causes tests to be marked as failed because of exception in log org.apache.geode.management.internal.cli.CommandManagerJUnitTest$MockPluginCommandNotFound -- -# Mock Extension commands -org.apache.geode.internal.cache.extension.mock.MockExtensionCommands \ No newline at end of file diff --git a/geode-lucene/src/main/resources/META-INF/services/org.springframework.shell.core.CommandMarker b/geode-lucene/src/main/resources/META-INF/services/org.springframework.shell.core.CommandMarker deleted file mode 100644 index 0879e15732..0000000000 --- a/geode-lucene/src/main/resources/META-

INF/services/org.springframework.shell.core.CommandMarker +++ /dev/null @@ -1,2 +0,0 @@ -# Lucene Extensions command -org.apache.geode.cache.lucene.internal.cli.LuceneIndexCommands \ No newline at end of file ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

16. jdeppe-pivotal opened a new pull request #1042: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1042 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? - [x] 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. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
17. pdxrunner commented on a change in pull request #1042: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1042#discussion_r150296170 ########## File path: geode-core/src/main/java/org/apache/geode/distributed/ConfigurationProperties.java ########## @@ -1860,7 +1860,10 @@ * <U>Default</U>: <code>""</code> * </p> * <U>Since</U>: GemFire 8.0 + * + * @deprecated Since Geode 1.4 (See GEODE-3007) */ + @Deprecated() Review comment: All uses of this property have been removed in this change - is that going to be a breaking change anywhere? ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
18. pdxrunner commented on a change in pull request #1042: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1042#discussion_r150298594 ########## File path: geode-core/src/main/java/org/apache/geode/management/internal/cli/GfshParser.java ########## @@ -57,10 +57,14 @@ public GfshParser(CommandManager commandManager) { add(command); } - for (Converter<?> converter : commandManager.getConverters()) { + for (Converter converter : commandManager.getConverters()) { if (converter.getClass().isAssignableFrom(ArrayConverter.class)) { ArrayConverter arrayConverter = (ArrayConverter) converter; - arrayConverter.setConverters(new HashSet<>(commandManager.getConverters())); + HashSet<Converter<?>> converters = new HashSet<>(); + for (Converter c : commandManager.getConverters()) { + converters.add(c); + } + arrayConverter.setConverters(converters); Review comment: Surely the logic in these nested foreach's could be simplified. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
19. PurelyApplied commented on a change in pull request #1042: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1042#discussion_r150362381 ########## File path: geode-core/src/main/java/org/apache/geode/management/internal/cli/GfshParser.java ########## @@ -57,10 +57,14 @@ public GfshParser(CommandManager commandManager) { add(command); } - for (Converter<?> converter : commandManager.getConverters()) { + for (Converter converter : commandManager.getConverters()) { if (converter.getClass().isAssignableFrom(ArrayConverter.class)) { ArrayConverter arrayConverter = (ArrayConverter) converter; - arrayConverter.setConverters(new HashSet<>(commandManager.getConverters())); + HashSet<Converter<?>> converters = new HashSet<>(); + for (Converter c : commandManager.getConverters()) { + converters.add(c); + } + arrayConverter.setConverters(converters); Review comment: At the very least, that should be a `Set` and not a `HashSet` on the left. If we wanted to collapse this into a `stream`, we could. ``` Set<Converter<?>> converters = new HashSet<>(); for (Converter c : commandManager.getConverters()) { converters.add(c); } ``` It seems to me that this is just doing some strange workaround to get a set of `Converter` to a set of `Converter<?>`. That could be rewritten as ``` Set<Converter<?>> converters = commandManager.getConverters().stream().map(c -> (Converter<?>) c).collect(Collectors.toSet()); ``` But at that point, the whole loop would go from ``` for (Converter converter : commandManager.getConverters()) { if (converter.getClass().isAssignableFrom(ArrayConverter.class)) { ArrayConverter arrayConverter = (ArrayConverter) converter; Set<Converter<?>> converters = commandManager.getConverters().stream().map(c -> (Converter<?>) c).collect(Collectors.toSet()); arrayConverter.setConverters(converters); } add(converter); } ``` to ``` commandManager.getConverters()

.stream() .filter(converter -> converter.getClass().isAssignableFrom(ArrayConverter.class)) .forEach(converter -> ((ArrayConverter) converter).setConverters(commandManager.getConverters().stream().map(c -> (Converter<?>) c).collect(Collectors.toSet()))); commandManager.getConverters().forEach(this::add); ``` with the caveat that the `add(converter)` is now happening for each after the others have been addressed, and not in the middle of the loop. But I don't think that should really matter, should it..? Looking at that, I don't really believe it to be more readable than the first code block in this comment. So probably split the difference, with ``` for (Converter converter : commandManager.getConverters()) { if (converter.getClass().isAssignableFrom(ArrayConverter.class)) { ArrayConverter arrayConverter = (ArrayConverter) converter; arrayConverter.setConverters(commandManager.getConverters().stream() .map(c -> (Converter<?>) c).collect(Collectors.toSet())); } add(converter); } ``` I don't know. I guess you could one step farther and reduce to ``` for (Converter converter : commandManager.getConverters()) { if (converter.getClass().isAssignableFrom(ArrayConverter.class)) { ((ArrayConverter) converter).setConverters(commandManager.getConverters().stream() .map(c -> (Converter<?>) c).collect(Collectors.toSet())); } add(converter); } ``` but I personally don't like one-lining casts and method calls. ... This comment got a lot bigger than I originally intended... ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

20. PurelyApplied commented on a change in pull request #1042: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1042#discussion_r150362984 ########## File path: geode-core/src/main/java/org/apache/geode/management/internal/cli/GfshParser.java ########## @@ -57,10 +57,14 @@ public GfshParser(CommandManager commandManager) { add(command); } - for (Converter<?> converter : commandManager.getConverters()) { + for (Converter converter : commandManager.getConverters()) { if (converter.getClass().isAssignableFrom(ArrayConverter.class)) { ArrayConverter arrayConverter = (ArrayConverter) converter; - arrayConverter.setConverters(new HashSet<>(commandManager.getConverters())); + HashSet<Converter<?>> converters = new HashSet<>(); + for (Converter c : commandManager.getConverters()) { + converters.add(c); + } + arrayConverter.setConverters(converters); Review comment: Also, just for the record: all of the above is predicated on a hazy understanding of the difference between `Converter` and `Converter<?>`. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
21. jdeppe-pivotal commented on a change in pull request #1042: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1042#discussion_r150558119 ########## File path: geode-core/src/main/java/org/apache/geode/management/internal/cli/GfshParser.java ########## @@ -57,10 +57,14 @@ public GfshParser(CommandManager commandManager) { add(command); } - for (Converter<?> converter : commandManager.getConverters()) { + for (Converter converter : commandManager.getConverters()) { if (converter.getClass().isAssignableFrom(ArrayConverter.class)) { ArrayConverter arrayConverter = (ArrayConverter) converter; - arrayConverter.setConverters(new HashSet<>(commandManager.getConverters())); + HashSet<Converter<?>> converters = new HashSet<>(); + for (Converter c : commandManager.getConverters()) { + converters.add(c); + } + arrayConverter.setConverters(converters); Review comment: The need here appears to stem from the way generics are being used (possibly incorrectly). Ideally we should just be able to say `HashSet<Converter<?>> converters = new HashSet<>(commandManager.getConverters();`. However that doesn't work because `getConverters` returns a `Set<Converter>` and not a `Set<Converter<?>>`. Ultimately the `setConverter` call wants to take a `Set<Converter<?>>` as an argument. Hence the need for this inner loop to keep the compiler happy. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
22. jdeppe-pivotal commented on a change in pull request #1042: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1042#discussion_r150558816 ########## File path: geode-core/src/main/java/org/apache/geode/distributed/ConfigurationProperties.java ########## @@ -1860,7 +1860,10 @@ * <U>Default</U>: <code>""</code> * </p> * <U>Since</U>: GemFire 8.0 + * * + * @deprecated Since Geode 1.4 (See GEODE-3007) */ + @Deprecated() Review comment: No, because the property itself isn't gone. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
23. jdeppe-pivotal closed pull request #1042: GEODE-3007: Simplify support for custom GFSH commands URL: https://github.com/apache/geode/pull/1042 This is a PR merged from a forked repository. As GitHub

hides the original diff on merge, it is displayed below for the sake of provenance: As this is a foreign pull request (from a fork), the diff is supplied below (as it won't show otherwise due to GitHub magic): diff --git a/geode-assembly/build.gradle b/geode-assembly/build.gradle index 1f139a5ac9..4615a6b08d 100755 --- a/geode-assembly/build.gradle +++ b/geode-assembly/build.gradle @@ -198,14 +198,12 @@ def cp = { it.contains('snappy') || it.contains('jgroups') || it.contains('netty') || - // dependencies from geode-lucene it.contains('lucene-analyzers-common') || it.contains('lucene-core') || it.contains('lucene-queries') || it.contains('lucene-queryparser') || it.contains('lucene-analyzers-phonetic') || - // dependencies from geode-protobuf it.contains('protobuf-java') } diff --git a/geode-assembly/src/test/java/org/apache/geode/management/internal/cli/commands/StatusLocatorRealGfshTest.java b/geode-assembly/src/test/java/org/apache/geode/management/internal/cli/commands/StatusLocatorRealGfshTest.java index 7e7b2bdca8..94ddc8f322 100644 --- a/geode-assembly/src/test/java/org/apache/geode/management/internal/cli/commands/StatusLocatorRealGfshTest.java +++ b/geode-assembly/src/test/java/org/apache/geode/management/internal/cli/commands/StatusLocatorRealGfshTest.java @@ -30,7 +30,6 @@ @Test public void statusLocatorSucceedsWhenConnected() throws Exception { GfshScript.of("start locator --name=locator1").execute(gfshRule); - GfshScript.of("connect", "status locator --name=locator1").execute(gfshRule); } diff --git a/geode-core/src/main/java/org/apache/geode/distributed/ConfigurationProperties.java b/geode-core/src/main/java/org/apache/geode/distributed/ConfigurationProperties.java index 649bfad407..4944e984bd 100644 --- a/geode-core/src/main/java/org/apache/geode/distributed/ConfigurationProperties.java +++ b/geode-core/src/main/java/org/apache/geode/distributed/ConfigurationProperties.java @@ -1860,7 +1860,10 @@ * <U>Default</U>: <code>""</code> * </p> * <U>Since</U>: GemFire 8.0 + * + * @deprecated Since Geode 1.4 (See GEODE-3007) */ + @Deprecated() String USER_COMMAND_PACKAGES = "user-command-packages"; /** * The static String definition of the "off-heap-memory-size" property <a diff --git a/geode-core/src/main/java/org/apache/geode/distributed/internal/AbstractDistributionConfig.java b/geode-core/src/main/java/org/apache/geode/distributed/internal/AbstractDistributionConfig.java index 43c5cb48fe..7c5929c4a8 100644 --- a/geode-core/src/main/java/org/apache/geode/distributed/internal/AbstractDistributionConfig.java +++ b/geode-core/src/main/java/org/apache/geode/distributed/internal/AbstractDistributionConfig.java @@ -1018,9 +1018,6 @@ public static Class _getAttributeType(String attName) { m.put(GROUPS, "A comma separated list of all the groups this member belongs to." + " Defaults to \"\"."); - m.put(USER_COMMAND_PACKAGES, - "A comma separated list of the names of the packages containing classes that implement user commands."); m.put(JMX_MANAGER, "If true then this member is willing to be a jmx manager. Defaults to false except on a locator."); m.put(JMX_MANAGER_START, diff --git a/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfig.java b/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfig.java index eabe6ccee6..3aeb19fad2 100644 --- a/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfig.java +++ b/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfig.java @@ -387,34 +387,6 @@ File DEFAULT_DEPLOY_WORKING_DIR = new File(System.getProperty("user.dir")); /** - * Returns the value of the {@link ConfigurationProperties#USER_COMMAND_PACKAGES} property - */ - @ConfigAttributeGetter(name = USER_COMMAND_PACKAGES) - String getUserCommandPackages(); - - /** - * Sets the system's user command path. - * - * @throws IllegalArgumentException if the specified value is not acceptable. - * @throws org.apache.geode.UnmodifiableException if this attribute can not be modified. - * @throws org.apache.geode.GemFireIOException if the set failure is caused by an error when - * writing to the system's configuration file. - */ - @ConfigAttributeSetter(name = USER_COMMAND_PACKAGES) - void setUserCommandPackages(String value); - - /** - * The name of the {@link ConfigurationProperties#USER_COMMAND_PACKAGES} property. - */ - @ConfigAttribute(type = String.class) - String USER_COMMAND_PACKAGES_NAME = USER_COMMAND_PACKAGES; - - /** - * The default value of the {@link ConfigurationProperties#USER_COMMAND_PACKAGES} property - */ - String DEFAULT_USER_COMMAND_PACKAGES = ""; - /** * Returns the value of the {@link ConfigurationProperties#LOG_FILE} property * * @return <code>null</code> if logging information goes to standard out diff --git a/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfigImpl.java b/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfigImpl.java index cceeaf9eb0..0c13480702 100644 --- a/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfigImpl.java +++ b/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfigImpl.java @@ -625,8 +625,6

@@ private Map<String, ConfigSource> sourceMap = Collections.synchronizedMap(new HashMap<String, ConfigSource>()); - protected String userCommandPackages = DEFAULT_USER_COMMAND_PACKAGES; - /** * "off-heap-memory-size" with value of "" or "<size> [g|m]" */ @@ -759,7 +757,6 @@ public DistributionConfigImpl(DistributionConfig other) { this.redisPort = other.getRedisPort(); this.redisBindAddress = other.getRedisBindAddress(); this.redisPassword = other.getRedisPassword(); - this.userCommandPackages = other.getUserCommandPackages(); // following added for 8.0 this.enableSharedConfiguration = other.getEnableClusterConfiguration(); @@ -1834,10 +1831,6 @@ public int getAsyncMaxQueueSize() { return this.asyncMaxQueueSize; } - public String getUserCommandPackages() { - return this.userCommandPackages; - } - public int getHttpServicePort() { return this.httpServicePort; } @@ -1862,10 +1855,6 @@ public void setStartDevRestApi(boolean value) { this.startDevRestApi = value; } - public void setUserCommandPackages(String value) { - this.userCommandPackages = value; - } - public boolean getDeltaPropagation() { return this.deltaPropagation; } @@ -3013,9 +3002,8 @@ public boolean equals(final Object o) { .append(sslTrustStore, that.sslTrustStore) .append(sslTrustStorePassword, that.sslTrustStorePassword) .append(locatorSSLAlias, that.locatorSSLAlias).append(sslDefaultAlias, that.sslDefaultAlias) - .append(sourceMap, that.sourceMap).append(userCommandPackages, that.userCommandPackages) - .append(offHeapMemorySize, that.offHeapMemorySize).append(shiroInit, that.shiroInit) - .isEquals(); + .append(sourceMap, that.sourceMap).append(offHeapMemorySize, that.offHeapMemorySize) + .append(shiroInit, that.shiroInit).isEquals(); } /** @@ -3083,9 +3071,8 @@ public int hashCode() { .append(sslCiphers).append(sslRequireAuthentication).append(sslKeyStore) .append(sslKeyStoreType).append(sslKeyStorePassword).append(sslTrustStore) .append(sslTrustStorePassword).append(sslWebServiceRequireAuthentication) - .append(locatorSSLAlias).append(sslDefaultAlias).append(sourceMap) - .append(userCommandPackages).append(offHeapMemorySize).append(lockMemory).append(shiroInit) - .append(modifiable).toHashCode(); + .append(locatorSSLAlias).append(sslDefaultAlias).append(sourceMap).append(offHeapMemorySize) + .append(lockMemory).append(shiroInit).append(modifiable).toHashCode(); } /** diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/beans/MemberMBeanBridge.java b/geode-core/src/main/java/org/apache/geode/management/internal/beans/MemberMBeanBridge.java index 0afdfd16db..de466af3db 100644 --- a/geode-core/src/main/java/org/apache/geode/management/internal/beans/MemberMBeanBridge.java +++ b/geode-core/src/main/java/org/apache/geode/management/internal/beans/MemberMBeanBridge.java @@ -338,8 +338,7 @@ public MemberMBeanBridge(InternalCache cache, SystemManagementService service) { this.config = system.getConfig(); try { - this.commandProcessor = - new OnlineCommandProcessor(system.getProperties(), cache.getSecurityService()); + this.commandProcessor = new OnlineCommandProcessor(cache.getSecurityService()); } catch (Exception e) { commandServiceInitError = e.getMessage(); logger.info(LogMarker.CONFIG, "Command processor could not be initialized. {}", diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/CommandManager.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/CommandManager.java index 44617d3f18..0b13d62cf4 100644 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/CommandManager.java +++ b/geode-core/src/main/java/org/apache/geode/management/internal/cli/CommandManager.java @@ -14,293 +14,93 @@ */ package org.apache.geode.management.internal.cli; -import static org.apache.geode.distributed.ConfigurationProperties.USER_COMMAND_PACKAGES; -import java.lang.reflect.Method; -import java.util.ArrayList; -import java.util.HashSet; -import java.util.Iterator; -import java.util.List; -import java.util.Properties; -import java.util.ServiceConfigurationError; -import java.util.ServiceLoader; +import static java.util.stream.Collectors.toSet; + +import java.util.Objects; import java.util.Set; -import java.util.StringTokenizer; +import java.util.stream.Stream; -import org.springframework.shell.converters.EnumConverter; -import org.springframework.shell.converters.SimpleFileConverter; import org.springframework.shell.core.CommandMarker; -import org.springframework.shell.core.Converter; -import org.springframework.shell.core.MethodTarget; -import org.springframework.shell.core.annotation.CliAvailabilityIndicator; -import org.springframework.shell.core.annotation.CliCommand; -import org.apache.geode.distributed.ConfigurationProperties; -import org.apache.geode.distributed.internal.DistributionConfig; -import org.apache.geode.internal.ClassPathLoader; -import org.apache.geode.management.internal.cli.commands.GfshCommand; -import org.apache.geode.management.internal.cli.help.Helper; -import org.apache.geode.management.internal.cli.shell.Gfsh; -import org.apache.geode.management.internal.cli.util.ClasspathScanLoadHelper; /** - * * this only takes care of

loading all available command markers and converters from the application * * @since GemFire 7.0 */ public class CommandManager { - public static final String USER_CMD_PACKAGES_PROPERTY = - DistributionConfig.GEMFIRE_PREFIX + USER_COMMAND_PACKAGES; - public static final String USER_CMD_PACKAGES_ENV_VARIABLE = "GEMFIRE_USER_COMMAND_PACKAGES"; - private static final Object INSTANCE_LOCK = new Object(); - - private final Helper helper = new Helper(); - - private final List<Converter<?>> converters = new ArrayList<Converter<?>>(); - private final List<CommandMarker> commandMarkers = new ArrayList<>(); + // Skip some of the Converters from Spring Shell for our customization + private static final Set<String> EXCLUDED_CLASSES = + Stream.of("-org.springframework.shell.converters.SimpleFileConverter", + "-org.springframework.shell.converters.FileConverter", + "-org.springframework.shell.converters.EnumConverter", + "-org.springframework.shell.commands.ExitCommands", + "-org.springframework.shell.commands.HelpCommands", + "-org.springframework.shell.commands.VersionCommands", + "-org.springframework.shell.commands.ConsoleCommands").collect(toSet()); + + private final LogWrapper logWrapper = LogWrapper.getInstance(); + private final Set<CommandMarker> commandMarkers; + private final Set<Converter> converters; + private final Helper helper; - private Properties cacheProperties; - private LogWrapper logWrapper; - - /** - * this constructor is used from Gfsh VM. We are getting the user-command-package from system - * environment. used by Gfsh. - */ public CommandManager() { - this(null); + helper = new Helper(); + converters = loadConverters(); + commandMarkers = loadCommandMarkers(); } - /** - * this is used when getting the instance in a cache server. We are getting the - * user-command-package from distribution properties. used by OnlineCommandProcessor. - */ - public CommandManager(final Properties cacheProperties) { - if (cacheProperties != null) { - this.cacheProperties = cacheProperties; - } - logWrapper = LogWrapper.getInstance(); - loadCommands(); - } + private Set<Converter> loadConverters() { + Set<Converter> converters = instantiateAllClassesImplementing(Converter.class); + raiseExceptionIfEmpty(converters, "converters"); - private static void raiseExceptionIfEmpty(Set<Class<?>> foundClasses, String errorFor) - throws IllegalStateException { - if (foundClasses == null || foundClasses.isEmpty()) { - throw new IllegalStateException(- "Required " + errorFor + " classes were not loaded. Check logs for errors."); - } + converters.forEach(this::setContextIfCommandManagerAware); + return converters; } - private void loadUserCommands() { - final Set<String> userCommandPackages = new HashSet<String>(); - - // Find by packages specified by the system property - if (System.getProperty(USER_CMD_PACKAGES_PROPERTY) != null) { - StringTokenizer tokenizer = - new StringTokenizer(System.getProperty(USER_CMD_PACKAGES_PROPERTY), ","); - while (tokenizer.hasMoreTokens()) { - userCommandPackages.add(tokenizer.nextToken()); - } - } - - // Find by packages specified by the environment variable - if (System.getenv().containsKey(USER_CMD_PACKAGES_ENV_VARIABLE)) { - StringTokenizer tokenizer = - new StringTokenizer(System.getenv().get(USER_CMD_PACKAGES_ENV_VARIABLE), ","); - while (tokenizer.hasMoreTokens()) { - userCommandPackages.add(tokenizer.nextToken()); - } - } + private Set<CommandMarker> loadCommandMarkers() { + Set<CommandMarker> commandMarkers = instantiateAllClassesImplementing(CommandMarker.class); + raiseExceptionIfEmpty(commandMarkers, "commandMarkers"); - // Find by packages specified in the distribution config - if (this.cacheProperties != null) { - String cacheUserCmdPackages = - this.cacheProperties.getProperty(ConfigurationProperties.USER_COMMAND_PACKAGES); - if (cacheUserCmdPackages != null && !cacheUserCmdPackages.isEmpty()) { - StringTokenizer tokenizer = new StringTokenizer(cacheUserCmdPackages, ","); - while (tokenizer.hasMoreTokens()) { - userCommandPackages.add(tokenizer.nextToken()); - } - } - } - - // Load commands found in all of the packages - for (String userCommandPackage : userCommandPackages) { - try { - Set<Class<?>> foundClasses = ClasspathScanLoadHelper - .scanPackageForClassesImplementing(userCommandPackage, CommandMarker.class); - for (Class<?> klass : foundClasses) { - try { - add((CommandMarker) klass.newInstance()); - } catch (Exception e) { - logWrapper.warning("Could not load User Commands from: " + klass + " due to " - + e.getLocalizedMessage()); // continue - } - } - raiseExceptionIfEmpty(foundClasses, "User Command"); - } catch (IllegalStateException e) { - logWrapper.warning(e.getMessage(), e); - throw e; - } - } - - /** - * Loads commands via {@link ServiceLoader} from {@link ClassPathLoader}. - * - * @since GemFire 8.1 - */ - private void loadPluginCommands() { - final Iterator<CommandMarker> iterator = ServiceLoader - .load(CommandMarker.class, ClassPathLoader.getLatest().asClassLoader()).iterator(); - while (iterator.hasNext()) { - try { - final CommandMarker commandMarker = iterator.next(); - try { - add(commandMarker); - } catch (Exception e) { - logWrapper.warning("Could not load Command from: " + commandMarker.getClass() + " due to " - + e.getLocalizedMessage(), e); // continue - } - } catch (ServiceConfigurationError e) { - logWrapper.severe("Could not load Command: " + e.getLocalizedMessage(), e); // continue - } - } +

commandMarkers.forEach(this::setContextIfCommandManagerAware); + commandMarkers.forEach(helper::registerCommand); + return commandMarkers; } + private <T> Set<T> instantiateAllClassesImplementing(Class<T> implementedInterface) { + Set<Class<? extends T>> classes = ClasspathScanLoadHelper.scanClasspathForClassesImplementing(+ implementedInterface, EXCLUDED_CLASSES.toArray(new String[0])); - private void loadCommands() { - loadUserCommands(); - - loadPluginCommands(); - loadGeodeCommands(); - loadConverters(); + return classes.stream().map(this::instantiateClass).filter(Objects::nonNull).collect(toSet()); } - private void loadConverters() { - Set<Class<?>> foundClasses; - // Converters + private <T> T instantiateClass(Class<T> classToInstantiate) { try { - foundClasses = ClasspathScanLoadHelper.scanPackageForClassesImplementing(- "org.apache.geode.management.internal.cli.converters", Converter.class); - for (Class<?> klass : foundClasses) { - try { - Converter<?> object = (Converter<?>) klass.newInstance(); - add(object); - - } catch (Exception e) { - logWrapper.warning(- "Could not load Converter from: " + klass + " due to " + e.getLocalizedMessage()); // continue - } - } - raiseExceptionIfEmpty(foundClasses, "Converters"); - - // Spring shell's converters - foundClasses = ClasspathScanLoadHelper.scanPackageForClassesImplementing(- "org.springframework.shell.converters", Converter.class); - for (Class<?> klass : foundClasses) { - if (!SHL_CONVERTERS_TOSKIP.contains(klass)) { - try { - add((Converter<?>) klass.newInstance()); - } catch (Exception e) { - logWrapper.warning(- "Could not load Converter from: " + klass + " due to " + e.getLocalizedMessage()); // continue - } - } - } - raiseExceptionIfEmpty(foundClasses, "Basic Converters"); - } catch (IllegalStateException e) { - logWrapper.warning(e.getMessage(), e); - throw e; + return classToInstantiate.newInstance(); + } catch (Exception e) { + logWrapper.warning("Could not load command or converter from: " + classToInstantiate, e); } + return null; } - private void loadGeodeCommands() { - // CommandMarkers - Set<Class<?>> foundClasses; - try { - // geode's commands - foundClasses = ClasspathScanLoadHelper.scanPackageForClassesImplementing(- GfshCommand.class.getPackage().getName(), CommandMarker.class); - - for (Class<?> klass : foundClasses) { - try { - add((CommandMarker) klass.newInstance()); - } catch (Exception e) { - logWrapper.warning(- "Could not load Command from: " + klass + " due to " + e.getLocalizedMessage()); // continue - } - } - raiseExceptionIfEmpty(foundClasses, "Commands"); - - // do not add Spring shell's commands for now. When we add it, we need to tell the parser that - // these are offline commands. - } catch (IllegalStateException e) { - logWrapper.warning(e.getMessage(), e); - throw e; + private void setContextIfCommandManagerAware(Object commandOrConverter) { + if (CommandManagerAware.class.isAssignableFrom(commandOrConverter.getClass())) { + ((CommandManagerAware) commandOrConverter).setCommandManager(this); } } - /** Skip some of the Converters from Spring Shell for our customization */ - private static List<Class> SHL_CONVERTERS_TOSKIP = new ArrayList(); - static { - // skip springs SimpleFileConverter to use our own FilePathConverter - SHL_CONVERTERS_TOSKIP.add(SimpleFileConverter.class); - // skip spring's EnumConverter to use our own EnumConverter - SHL_CONVERTERS_TOSKIP.add(EnumConverter.class); - } - - public List<Converter<?>> getConverters() { - return converters; - } - - public List<CommandMarker> getCommandMarkers() { - return commandMarkers; - } - - /** - * Method to add new Converter - * - * @param converter - */ - void add(Converter<?> converter) { - if (CommandManagerAware.class.isAssignableFrom(converter.getClass())) { - ((CommandManagerAware) converter).setCommandManager(this); - } - converters.add(converter); - } - - /** - * Method to add new Commands to the parser - * - * @param commandMarker - */ - void add(CommandMarker commandMarker) { - if (CommandManagerAware.class.isAssignableFrom(commandMarker.getClass())) { - ((CommandManagerAware) commandMarker).setCommandManager(this); - } - commandMarkers.add(commandMarker); - for (Method method : commandMarker.getClass().getMethods()) { - CliCommand cliCommand = method.getAnnotation(CliCommand.class); - CliAvailabilityIndicator availability = method.getAnnotation(CliAvailabilityIndicator.class); - if (cliCommand == null && availability == null) { - continue; - } - - if (cliCommand != null) { - helper.addCommand(cliCommand, method); - } - - if (availability != null) { - helper.addAvailabilityIndicator(availability, new MethodTarget(method, commandMarker)); - } + private static void raiseExceptionIfEmpty(Set<?> foundClasses, String classType) + throws IllegalStateException { + if (foundClasses == null || foundClasses.isEmpty()) { + throw new IllegalStateException("No " + classType + " were loaded. Check logs for errors."); } } - public Helper getHelper() { - return helper; - } - public String obtainHelp(String buffer) { int terminalWidth = -1; Gfsh gfsh = Gfsh.getCurrentInstance(); @@ -314,4 +114,15 @@ public String obtainHint(String topic) { return helper.getHint(topic); } + public Set<Converter> getConverters() { + return converters; + } + + public Set<CommandMarker> getCommandMarkers() { + return commandMarkers; + } + + public Helper getHelper() { + return helper; + } } diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/GfshParser.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/GfshParser.java index 9cebc63c34..179d4ec44b 100755 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/GfshParser.java +++ b/geode-

core/src/main/java/org/apache/geode/management/internal/cli/GfshParser.java @@ -57,10 +57,14 @@ public GfshParser(CommandManager commandManager) { add(command); } - for (Converter<?> converter : commandManager.getConverters()) { + for (Converter converter : commandManager.getConverters()) { if (converter.getClass().isAssignableFrom(ArrayConverter.class)) { ArrayConverter arrayConverter = (ArrayConverter) converter; - arrayConverter.setConverters(new HashSet<> (commandManager.getConverters())); + HashSet<Converter<?>> converters = new HashSet<>(); + for (Converter c : commandManager.getConverters()) { + converters.add(c); + } + arrayConverter.setConverters(converters); } add(converter); } diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/help/Helper.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/help/Helper.java index 917f3e4961..ec6f4e3ae1 100644 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/help/Helper.java +++ b/geode-core/src/main/java/org/apache/geode/management/internal/cli/help/Helper.java @@ -14,9 +14,10 @@ */ package org.apache.geode.management.internal.cli.help; +import static java.util.stream.Collectors.joining; + import java.lang.annotation.Annotation; import java.lang.reflect.Method; -import java.util.ArrayList; import java.util.Arrays; import java.util.Collection; import java.util.Collections; @@ -27,6 +28,7 @@ import java.util.TreeMap; import org.apache.commons.lang.StringUtils; +import org.springframework.shell.core.CommandMarker; import org.springframework.shell.core.MethodTarget; import org.springframework.shell.core.annotation.CliAvailabilityIndicator; import org.springframework.shell.core.annotation.CliCommand; @@ -65,8 +67,7 @@ private final Map<String, Topic> topics = new HashMap<>(); private final Map<String, Method> commands = new TreeMap<String, Method>(); - private final Map<String, MethodTarget> availabilityIndicators = - new HashMap<String, MethodTarget>(); + private final Map<String, MethodTarget> availabilityIndicators = new HashMap<>(); public Helper() { initTopic(CliStrings.DEFAULT_TOPIC_GEODE, CliStrings.DEFAULT_TOPIC_GEODE__DESC); @@ -94,7 +95,25 @@ private void initTopic(String topic, String desc) { topics.put(topic, new Topic(topic, desc)); } - public void addCommand(CliCommand command, Method commandMethod) { + public void registerCommand(CommandMarker commandMarker) { + for (Method method : commandMarker.getClass().getMethods()) { + CliCommand cliCommand = method.getAnnotation(CliCommand.class); + CliAvailabilityIndicator availability = method.getAnnotation(CliAvailabilityIndicator.class); + if (cliCommand == null && availability == null) { + continue; + } + + if (cliCommand != null) { + addCommand(cliCommand, method); + } + + if (availability != null) { + addAvailabilityIndicator(availability, new MethodTarget(method, commandMarker)); + } + } + } + + protected void addCommand(CliCommand command, Method commandMethod) { // put all the command synonyms in the command map Arrays.stream(command.value()).forEach(cmd -> { commands.put(cmd, commandMethod); @@ -120,7 +139,8 @@ public void addCommand(CliCommand command, Method commandMethod) { }); } - public void addAvailabilityIndicator(CliAvailabilityIndicator availability, + private void addAvailabilityIndicator(CliAvailabilityIndicator availability, + MethodTarget target) { Arrays.stream(availability.value()).forEach(command -> { availabilityIndicators.put(command, target); }); @@ -152,9 +172,7 @@ public String getHint(String buffer) { builder.append(CliStrings.HINT__MSG__TOPICS_AVAILABLE).append(GfshParser.LINE_SEPARATOR) .append(GfshParser.LINE_SEPARATOR); - List<String> sortedTopics = new ArrayList<>(topics.keySet()); - Collections.sort(sortedTopics); - sortedTopics.stream() + topics.keySet().stream().sorted() .forEachOrdered(topic -> builder.append(topic).append(GfshParser.LINE_SEPARATOR)); return builder.toString(); } @@ -165,8 +183,7 @@ public String getHint(String buffer) { } builder.append(topic.desc).append(GfshParser.LINE_SEPARATOR).append(GfshParser.LINE_SEPARATOR); - Collections.sort(topic.relatedCommands); - topic.relatedCommands.stream().forEachOrdered(command -> + topic.relatedCommands.stream().sorted().forEach(command -> builder.append(command.command).append(": ").append(command.desc).append(GfshParser.LINE_SEPARATOR)); return builder.toString(); } @@ -175,7 +192,7 @@ public String getHint(String buffer) { return topics.keySet(); } - boolean isAvailable(String command) { + private boolean isAvailable(String command) { MethodTarget target = availabilityIndicators.get(command); if (target == null) { return true; @@ -261,16 +278,12 @@ HelpBlock getOptionDetail(CliOption cliOption) { HelpBlock optionNode = new HelpBlock(getPrimaryKey(cliOption)); String help = cliOption.help(); optionNode.addChild(new HelpBlock((StringUtils.isNotBlank(help) ? help : ""))); - if (getSynonyms(cliOption).size() > 0) { - StringBuilder builder = new StringBuilder(); - for (String string : getSynonyms(cliOption)) { - if (builder.length() > 0) { - builder.append(","); - } - builder.append(string); - } - optionNode.addChild(new HelpBlock(SYNONYMS_SUB_NAME + builder.toString())); + + String synonyms = getSynonyms(cliOption).stream().collect(joining(",")); + if (StringUtils.isNotEmpty(synonyms)) { + optionNode.addChild(new HelpBlock(SYNONYMS_SUB_NAME + synonyms)); } + optionNode.addChild(

new HelpBlock(REQUIRED_SUB_NAME + ((cliOption.mandatory()) ? TRUE_TOKEN : FALSE_TOKEN))); if (!isNullOrBlank(cliOption.specifiedDefaultValue())) { @@ -359,18 +372,14 @@ private static String getPrimaryKey(CliOption option) { } private static List<String> getSynonyms(CliOption option) { - List<String> synonyms = new ArrayList<>(); String[] keys = option.key(); if (keys.length < 2) - return synonyms; + return Collections.emptyList(); // if the primary key is empty (like sh and help command), then there should be no synonyms. if ("".equals(keys[0])) - return synonyms; + return Collections.emptyList(); - for (int i = 1; i < keys.length; i++) { - synonyms.add(keys[i]); - } - return synonyms; + return Arrays.asList(keys).subList(1, keys.length); } private static boolean isNullOrBlank(String value) { diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/MemberCommandService.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/MemberCommandService.java index 6130117d2e..a4c6f59802 100644 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/MemberCommandService.java +++ b/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/MemberCommandService.java @@ -35,8 +35,7 @@ public MemberCommandService(InternalCache cache) throws CommandServiceException { this.cache = cache; try { - this.onlineCommandProcessor = new OnlineCommandProcessor(- cache.getDistributedSystem().getProperties(), cache.getSecurityService()); + this.onlineCommandProcessor = new OnlineCommandProcessor(cache.getSecurityService()); } catch (Exception e) { throw new CommandServiceException("Could not load commands.", e); } diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessor.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessor.java index 7fa4acba51..38ab364392 100644 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessor.java +++ b/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessor.java @@ -18,7 +18,6 @@ import java.lang.reflect.Method; import java.util.Collections; import java.util.Map; -import java.util.Properties; import org.springframework.shell.core.Parser; import org.springframework.shell.event.ParseResult; @@ -26,7 +25,6 @@ import org.apache.geode.annotations.TestingOnly; import org.apache.geode.internal.security.SecurityService; -import org.apache.geode.internal.security.SecurityServiceFactory; import org.apache.geode.management.cli.CliMetaData; import org.apache.geode.management.cli.CommandProcessingException; import org.apache.geode.management.cli.Result; @@ -49,20 +47,14 @@ private final SecurityService securityService; - @TestingOnly - public OnlineCommandProcessor() throws ClassNotFoundException, IOException { - this(new Properties(), SecurityServiceFactory.create()); - } - - public OnlineCommandProcessor(Properties cacheProperties, SecurityService securityService) + public OnlineCommandProcessor(SecurityService securityService) throws ClassNotFoundException, IOException { - this(cacheProperties, securityService, new CommandExecutor()); + this(securityService, new CommandExecutor()); } @TestingOnly - public OnlineCommandProcessor(Properties cacheProperties, SecurityService securityService, - CommandExecutor commandExecutor) { - this.gfshParser = new GfshParser(new CommandManager(cacheProperties)); + public OnlineCommandProcessor(SecurityService securityService, CommandExecutor commandExecutor) { + this.gfshParser = new GfshParser(new CommandManager()); this.executor = commandExecutor; this.securityService = securityService; } diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/util/ClasspathScanLoadHelper.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/util/ClasspathScanLoadHelper.java index 046d890567..11da4b208a 100644 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/util/ClasspathScanLoadHelper.java +++ b/geode-core/src/main/java/org/apache/geode/management/internal/cli/util/ClasspathScanLoadHelper.java @@ -21,6 +21,7 @@ import java.util.Set; import io.github.lukehutch.fastclasspathscanner.FastClasspathScanner; +import io.github.lukehutch.fastclasspathscanner.matchprocessor.ImplementingClassMatchProcessor; /** * Utility class to scan class-path & load classes. @@ -28,18 +29,26 @@ * @since GemFire 7.0 */ public class ClasspathScanLoadHelper { - public static Set<Class<?>> scanPackageForClassesImplementing(String packageToScan, - Class<?> implementedInterface) { - Set<Class<?>> classesImplementing = new HashSet<>(); - new FastClasspathScanner(packageToScan) - .matchClassesImplementing(implementedInterface, classesImplementing::add).scan(); + public static <T> Set<Class<? extends T>> scanClasspathForClassesImplementing(+ Class<T> implementedInterface, String... packageSpec) { + return scanPackageForClassesImplementing(implementedInterface, packageSpec); + } + + public static <T> Set<Class<? extends T>> scanPackageForClassesImplementing(Class<T> implementedInterface, String... packageSpec) { + Set<Class<? extends T>> classesImplementing = new

HashSet<>(); + ImplementingClassMatchProcessor<T> matchProcessor = classesImplementing::add; + + new FastClasspathScanner(packageSpec) + .matchClassesImplementing(implementedInterface, matchProcessor).scan(); return classesImplementing.stream().filter(ClasspathScanLoadHelper::isInstantiable) .collect(toSet()); } - private static boolean isInstantiable(Class<?> klass) { - int modifiers = klass.getModifiers(); + private static <T> boolean isInstantiable(Class<T> classToInstantiate) { + int modifiers = classToInstantiate.getModifiers(); return !Modifier.isAbstract(modifiers) && !Modifier.isInterface(modifiers) && Modifier.isPublic(modifiers); diff --git a/geode-core/src/test/java/org/apache/geode/distributed/internal/DistributionConfigJUnitTest.java b/geode-core/src/test/java/org/apache/geode/distributed/internal/DistributionConfigJUnitTest.java index 7e5f83f4f2..e874898d82 100644 --- a/geode-core/src/test/java/org/apache/geode/distributed/internal/DistributionConfigJUnitTest.java +++ b/geode-core/src/test/java/org/apache/geode/distributed/internal/DistributionConfigJUnitTest.java @@ -99,7 +99,7 @@ public void before() { @Test public void testGetAttributeNames() { String[] attNames = AbstractDistributionConfig._getAttNames(); - assertEquals(attNames.length, 157); + assertEquals(156, attNames.length); List boolList = new ArrayList(); List intList = new ArrayList(); @@ -135,7 +135,7 @@ public void testGetAttributeNames() { // are. assertEquals(29, boolList.size()); assertEquals(33, intList.size()); - assertEquals(86, stringList.size()); + assertEquals(85, stringList.size()); assertEquals(5, fileList.size()); assertEquals(4, otherList.size()); } diff --git a/geode-core/src/test/java/org/apache/geode/management/internal/cli/ClasspathScanLoadHelperJUnitTest.java b/geode-core/src/test/java/org/apache/geode/management/internal/cli/ClasspathScanLoadHelperJUnitTest.java index 1dfd61d9a1..e48c8d24a7 100644 --- a/geode-core/src/test/java/org/apache/geode/management/internal/cli/ClasspathScanLoadHelperJUnitTest.java +++ b/geode-core/src/test/java/org/apache/geode/management/internal/cli/ClasspathScanLoadHelperJUnitTest.java @@ -14,7 +14,8 @@ */ package org.apache.geode.management.internal.cli; -import static org.junit.Assert.*; +import static org.junit.Assert.assertEquals; +import static org.junit.Assert.assertTrue; import java.util.Set; @@ -35,36 +36,36 @@ private final String PACKAGE_NAME = "org.apache.geode.management.internal.cli.domain"; private final String WRONG_PACKAGE_NAME = "org.apache.geode.management.internal.cli.domain1"; - private final Class<?> INTERFACE1 = Interface1.class; - private final Class<?> NO_IMPL_INTERFACE = Versionable.class; - private final Class<?> INTERFACE2 = Interface2.class; - private final Class<?> IMPL1 = Impl1.class; - private final Class<?> IMPL2 = Impl12.class; - private final Class<?> ABSTRACT_IMPL = AbstractImpl.class; + private final Class INTERFACE1 = Interface1.class; + private final Class NO_IMPL_INTERFACE = Versionable.class; + private final Class INTERFACE2 = Interface2.class; + private final Class IMPL1 = Impl1.class; + private final Class IMPL2 = Impl12.class; + private final Class ABSTRACT_IMPL = AbstractImpl.class; @Test public void testLoadAndGet() throws Exception { Set<Class<?>> classLoaded = - ClasspathScanLoadHelper.scanPackageForClassesImplementing(PACKAGE_NAME, INTERFACE1); + ClasspathScanLoadHelper.scanPackageForClassesImplementing(INTERFACE1, PACKAGE_NAME); assertEquals(2, classLoaded.size()); assertTrue(classLoaded.contains(IMPL1)); assertTrue(classLoaded.contains(IMPL2)); classLoaded = - ClasspathScanLoadHelper.scanPackageForClassesImplementing(PACKAGE_NAME, INTERFACE2); + ClasspathScanLoadHelper.scanPackageForClassesImplementing(INTERFACE2, PACKAGE_NAME); assertEquals(1, classLoaded.size()); assertTrue(classLoaded.contains(IMPL2)); classLoaded = - ClasspathScanLoadHelper.scanPackageForClassesImplementing(WRONG_PACKAGE_NAME, INTERFACE2); + ClasspathScanLoadHelper.scanPackageForClassesImplementing(INTERFACE2, WRONG_PACKAGE_NAME); assertEquals(0, classLoaded.size()); classLoaded = - ClasspathScanLoadHelper.scanPackageForClassesImplementing(PACKAGE_NAME, NO_IMPL_INTERFACE); + ClasspathScanLoadHelper.scanPackageForClassesImplementing(NO_IMPL_INTERFACE, PACKAGE_NAME); assertEquals(0, classLoaded.size()); - classLoaded = ClasspathScanLoadHelper.scanPackageForClassesImplementing(WRONG_PACKAGE_NAME, - NO_IMPL_INTERFACE); + classLoaded = ClasspathScanLoadHelper.scanPackageForClassesImplementing(NO_IMPL_INTERFACE, + WRONG_PACKAGE_NAME); assertEquals(0, classLoaded.size()); } } diff --git a/geode-core/src/test/java/org/apache/geode/management/internal/cli/CommandManagerJUnitTest.java b/geode-core/src/test/java/org/apache/geode/management/internal/cli/CommandManagerJUnitTest.java index 813347128c..c65ff04936 100644 --- a/geode-core/src/test/java/org/apache/geode/management/internal/cli/CommandManagerJUnitTest.java +++ b/geode-core/src/test/java/org/apache/geode/management/internal/cli/CommandManagerJUnitTest.java @@ -18,20 +18,12 @@ import static org.junit.Assert.assertNotNull; import static org.junit.Assert.assertTrue; -import java.util.Map; - -import org.apache.geode.management.internal.cli.CommandManager; -import org.apache.geode.management.internal.cli.CommandManagerJUnitTest; -import org.apache.geode.management.internal.cli.CommandManagerJUnitTest.TestCommands; -import org.apache.geode.management.internal.cli.CommandManagerJUnitTest.TestCommands2; -import org.apache.geode.management.internal.cli.CommandManagerJUnitTest.TestCommands3; -import org.apache.geode.management.internal.cli.CommandManagerJUnitTest.TestCommands4; import org.junit.Test;

java.util.List; - import org.junit.Before; import org.junit.Test; import org.junit.experimental.categories.Category; import org.springframework.shell.core.CommandMarker; -import org.springframework.shell.core.Completion; -import org.springframework.shell.core.Converter; -import org.springframework.shell.core.MethodTarget; -import org.springframework.shell.core.annotation.CliAvailabilityIndicator; import org.springframework.shell.core.annotation.CliCommand; -import org.springframework.shell.core.annotation.CliOption; -import org.apache.geode.management.cli.CliMetaData; import org.apache.geode.management.cli.Result; import org.apache.geode.management.internal.security.ResourceOperation; import org.apache.geode.security.ResourcePermission.Operation; @@ -43,49 +35,6 @@ */ @Category(UnitTest.class) public class CommandManagerJUnitTest { - - private static final String COMMAND1_NAME = "command1"; - private static final String COMMAND1_NAME_ALIAS = "command1_alias"; - private static final String COMMAND2_NAME = "c2"; - - private static final String COMMAND1_HELP = "help for " + COMMAND1_NAME; - // ARGUMENTS - private static final String ARGUMENT1_NAME = "argument1"; - private static final String ARGUMENT1_HELP = "help for argument1"; - private static final String ARGUMENT1_CONTEXT = "context for argument 1"; - private static final Completion[] ARGUMENT1_COMPLETIONS = - {new Completion("arg1"), new Completion("arg1alt")}; - private static final String ARGUMENT2_NAME = "argument2"; - private static final String ARGUMENT2_CONTEXT = "context for argument 2"; - private static final String ARGUMENT2_HELP = "help for argument2"; - private static final String ARGUMENT2_UNSPECIFIED_DEFAULT_VALUE = - "{unspecified default value for argument2}"; - private static final Completion[] ARGUMENT2_COMPLETIONS = - {new Completion("arg2"), new Completion("arg2alt")}; - - // OPTIONS - private static final String OPTION1_NAME = "option1"; - private static final String OPTION1_SYNONYM = "opt1"; - private static final String OPTION1_HELP = "help for option1"; - private static final String OPTION1_CONTEXT = "context for option1"; - private static final String OPTION1_SPECIFIED_DEFAULT_VALUE = - "{specified default value for option1}"; - private static final Completion[] OPTION1_COMPLETIONS = - {new Completion("option1"), new Completion("option1Alternate")}; - private static final String OPTION2_NAME = "option2"; - private static final String OPTION2_HELP = "help for option2"; - private static final String OPTION2_CONTEXT = "context for option2"; - private static final String OPTION2_SPECIFIED_DEFAULT_VALUE = - "{specified default value for option2}"; - private static final String OPTION3_NAME = "option3"; - private static final String OPTION3_SYNONYM = "opt3"; - private static final String OPTION3_HELP = "help for option3"; - private static final String OPTION3_CONTEXT = "context for option3"; - private static final String OPTION3_SPECIFIED_DEFAULT_VALUE = - "{specified default value for option3}"; - private static final String OPTION3_UNSPECIFIED_DEFAULT_VALUE = - "{unspecified default value for option3}"; - private CommandManager commandManager; @Before @@ -111,11 +60,6 @@ public void testCommandManagerInstance() throws Exception { assertNotNull(commandManager); } - /** - * Tests {@link CommandManager#loadPluginCommands()}. - * - * @since GemFire 8.1 - */ @Test public void testCommandManagerLoadPluginCommands() throws Exception { assertNotNull(commandManager); @@ -123,97 +67,10 @@ public void testCommandManagerLoadPluginCommands() throws Exception { assertNotNull(commandManager); @@ -123,97 +67,10 @@ public void testCommandManagerLoadPluginCommands() throws Exception { // see META-INF/services/org.springframework.shell.core.CommandMarker service loader file. assertTrue("Should find listed plugin.", commandManager.getHelper().getCommands().contains("mock plugin command")); - assertTrue("Should not find unlisted plugin.", - !commandManager.getHelper().getCommands().contains("mock plugin command unlisted")); - } - - /** - * class that represents dummy commands - */ - public static class Commands implements CommandMarker { - @CliCommand(value = {COMMAND1_NAME, COMMAND1_NAME_ALIAS}, help = COMMAND1_HELP) - @CliMetaData(shellOnly = true, relatedTopic = {"relatedTopicOfCommand1"}) - @ResourceOperation(resource = Resource.CLUSTER, operation = Operation.READ) - public static String command1(- @CliOption(key = ARGUMENT1_NAME, optionContext = ARGUMENT1_CONTEXT, help = ARGUMENT1_HELP, - mandatory = true) String argument1, - @CliOption(key = ARGUMENT2_NAME, optionContext = ARGUMENT2_CONTEXT, help = ARGUMENT2_HELP, - unspecifiedDefaultValue = ARGUMENT2_UNSPECIFIED_DEFAULT_VALUE) String argument2, - @CliOption(key = {OPTION1_NAME, OPTION1_SYNONYM}, help = OPTION1_HELP, mandatory = true, - optionContext = OPTION1_CONTEXT, - specifiedDefaultValue = OPTION1_SPECIFIED_DEFAULT_VALUE) String option1, - @CliOption(key = {OPTION2_NAME}, help = OPTION2_HELP, optionContext = OPTION2_CONTEXT, - specifiedDefaultValue = OPTION2_SPECIFIED_DEFAULT_VALUE) String option2, - @CliOption(key = {OPTION3_NAME, OPTION3_SYNONYM}, help = OPTION3_HELP, - optionContext = OPTION3_CONTEXT, - unspecifiedDefaultValue = OPTION3_UNSPECIFIED_DEFAULT_VALUE, - specifiedDefaultValue = OPTION3_SPECIFIED_DEFAULT_VALUE) String option3) { - return null; - } - - @CliCommand(value = {COMMAND2_NAME}) - @ResourceOperation(resource = Resource.CLUSTER, operation =

Operation.READ) - public static String command2() { - return null; - } - - @CliCommand(value = {"testParamConcat"}) - @ResourceOperation(resource = Resource.CLUSTER, operation = Operation.READ) - public static Result testParamConcat(@CliOption(key = {"string"}) String string, - @CliOption(key = {"stringArray"}) String[] stringArray, - @CliOption(key = {"integer"}) Integer integer, - @CliOption(key = {"colonArray"}) String[] colonArray) { - return null; - } - - @CliCommand(value = {"testMultiWordArg"}) - @ResourceOperation(resource = Resource.CLUSTER, operation = Operation.READ) - public static Result testMultiWordArg(@CliOption(key = "arg1") String arg1, - @CliOption(key = "arg2") String arg2) { - return null; - } - - @CliAvailabilityIndicator({COMMAND1_NAME}) - public boolean isAvailable() { - return true; // always available on server - } + assertThat(+ commandManager.getCommandMarkers().stream().anyMatch(c -> c instanceof MockPluginCommand)); } - /** - * Used by testCommandManagerLoadPluginCommands - */ - private static class SimpleConverter implements Converter<String> { - - @Override - public boolean supports(Class<?> type, String optionContext) { - return type.isAssignableFrom(String.class); - } - - @Override - public String convertFromText(String value, Class<?> targetType, String optionContext) { - return value; - } - - @Override - public boolean getAllPossibleValues(List<Completion> completions, Class<?> targetType, - String existingData, String context, MethodTarget target) { - if (context.equals(ARGUMENT1_CONTEXT)) { - for (Completion completion : ARGUMENT1_COMPLETIONS) { - completions.add(completion); - } - } else if (context.equals(ARGUMENT2_CONTEXT)) { - for (Completion completion : ARGUMENT2_COMPLETIONS) { - completions.add(completion); - } - } else if (context.equals(OPTION1_CONTEXT)) { - for (Completion completion : OPTION1_COMPLETIONS) { - completions.add(completion); - } - } - return true; - } - } public static class MockPluginCommand implements CommandMarker { @CliCommand(value = "mock plugin command") @@ -222,13 +79,4 @@ public Result mockPluginCommand() { return null; } } - - public static class MockPluginCommandUnlisted implements CommandMarker { - @CliCommand(value = "mock plugin command unlisted") - @ResourceOperation(resource = Resource.CLUSTER, operation = Operation.READ) - public Result mockPluginCommandUnlisted() { - return null; - } - } } diff --git a/geode-core/src/test/java/org/apache/geode/management/internal/cli/commands/CliCommandTestBase.java b/geode-core/src/test/java/org/apache/geode/management/internal/cli/commands/CliCommandTestBase.java index 5714829cfa..d79993201b 100644 --- a/geode-core/src/test/java/org/apache/geode/management/internal/cli/commands/CliCommandTestBase.java +++ b/geode-core/src/test/java/org/apache/geode/management/internal/cli/commands/CliCommandTestBase.java @@ -27,9 +27,9 @@ import java.io.PrintStream; import java.net.InetAddress; import java.net.UnknownHostException; -import java.util.List; import java.util.Map; import java.util.Properties; +import java.util.Set; import java.util.regex.Matcher; import java.util.regex.Pattern; @@ -80,7 +80,7 @@ public static boolean checkIfCommandsAreLoadedOrNot() { CommandManager manager = new CommandManager(); - List<CommandMarker> commands = manager.getCommandMarkers(); + Set<CommandMarker> commands = manager.getCommandMarkers(); return commands.size() >= 1; } diff --git a/geode-core/src/test/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessorTest.java b/geode-core/src/test/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessorTest.java index df00cf17aa..db7176ba4f 100644 --- a/geode-core/src/test/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessorTest.java +++ b/geode-core/src/test/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessorTest.java @@ -21,8 +21,6 @@ import static org.mockito.Mockito.mock; import static org.mockito.Mockito.when; -import java.util.Properties; - import org.junit.Before; import org.junit.Test; import org.junit.experimental.categories.Category; @@ -36,7 +34,6 @@ @Category(UnitTest.class) public class OnlineCommandProcessorTest { - Properties properties; SecurityService securityService; CommandExecutor executor; OnlineCommandProcessor onlineCommandProcessor; @@ -44,13 +41,12 @@ public class OnlineCommandProcessorTest { @Before public void before() { - properties = new Properties(); securityService = mock(SecurityService.class); executor = mock(CommandExecutor.class); result = mock(Result.class); when(executor.execute(any())).thenReturn(result); - onlineCommandProcessor = new OnlineCommandProcessor(properties, securityService, executor); + onlineCommandProcessor = new OnlineCommandProcessor(securityService, executor); } diff --git a/geode-core/src/test/resources/META-INF/services/org.springframework.shell.core.CommandMarker b/geode-core/src/test/resources/META-INF/services/org.springframework.shell.core.CommandMarker deleted file mode 100644 index 1bb7c6d141..0000000000 --- a/geode-core/src/test/resources/META-INF/services/org.springframework.shell.core.CommandMarker +++ /dev/null @@ -1,8 +0,0 @@ -# Mock command for

org.apache.geode.management.internal.cli.CommandManagerJUnitTest.testLoadPluginCommands - org.apache.geode.management.internal.cli.CommandManagerJUnitTest$MockPluginCommand -# Should cause ServiceConfigurationException while iterating. -#TODO jbarrett - causes tests to be marked as failed because of exception in log org.apache.geode.management.internal.cli.CommandManagerJUnitTest$MockPluginCommandNotFound - --# Mock Extension commands -org.apache.geode.internal.cache.extension.mock.MockExtensionCommands \ No newline at end of file diff --git a/geode-lucene/src/main/resources/META-INF/services/org.springframework.shell.core.CommandMarker b/geode-lucene/src/main/resources/META-INF/services/org.springframework.shell.core.CommandMarker deleted file mode 100644 index 0879e15732..0000000000 --- a/geode-lucene/src/main/resources/META-INF/services/org.springframework.shell.core.CommandMarker +++ /dev/null @@ -1,2 +0,0 @@ -# Lucene Extensions command -org.apache.geode.cache.lucene.internal.cli.LuceneIndexCommands \ No newline at end of file ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

24. Commit 0e5dd6ba45519463147337c8265db15e8f1840fc in geode's branch refs/heads/develop from [~jens.deppe] [https://gitbox.apache.org/repos/asf?p=geode.git;h=0e5dd6b] GEODE-3007: Simplify support for custom GFSH commands (#1042) * GEODE-3007: Simplify support for custom GFSH commands
25. Commit 0e5dd6ba45519463147337c8265db15e8f1840fc in geode's branch refs/heads/develop from [~jens.deppe] [https://gitbox.apache.org/repos/asf?p=geode.git;h=0e5dd6b] GEODE-3007: Simplify support for custom GFSH commands (#1042) * GEODE-3007: Simplify support for custom GFSH commands
26. Commit 0e5dd6ba45519463147337c8265db15e8f1840fc in geode's branch refs/heads/feature/GEODE-3940 from [~jens.deppe] [https://gitbox.apache.org/repos/asf?p=geode.git;h=0e5dd6b] GEODE-3007: Simplify support for custom GFSH commands (#1042) * GEODE-3007: Simplify support for custom GFSH commands
27. Commit 0e5dd6ba45519463147337c8265db15e8f1840fc in geode's branch refs/heads/feature/GEODE-3940 from [~jens.deppe] [https://gitbox.apache.org/repos/asf?p=geode.git;h=0e5dd6b] GEODE-3007: Simplify support for custom GFSH commands (#1042) * GEODE-3007: Simplify support for custom GFSH commands
28. jdeppe-pivotal opened a new pull request #1072: Revert "GEODE-3007: Simplify support for custom GFSH commands (#1042)" URL: https://github.com/apache/geode/pull/1072 This reverts commit 0e5dd6ba45519463147337c8265db15e8f1840fc. 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: - [ ] Is there a JIRA ticket associated with this PR? Is it referenced in the commit message? - [ ] Has your PR been rebased against the latest commit within the target branch (typically `develop`)? - [ ] Is your initial contribution a single, squashed commit? - [ ] 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. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
29. jdeppe-pivotal closed pull request #1072: Revert "GEODE-3007: Simplify support for custom GFSH commands (#1042)" URL: https://github.com/apache/geode/pull/1072 This is a PR merged from a forked repository. As GitHub hides the original diff on merge, it is displayed below for the sake of provenance: As this is a foreign pull request (from a fork), the diff is supplied below (as it won't show otherwise due to GitHub magic): diff --git a/geode-assembly/build.gradle b/geode-assembly/build.gradle index 4615a6b08d..1f139a5ac9 100755 --- a/geode-assembly/build.gradle +++ b/geode-assembly/build.gradle @@ -198,12 +198,14 @@ def cp = { it.contains('snappy') || it.contains('jgroups') || it.contains('netty') || + // dependencies from geode-lucene it.contains('lucene-analyzers-common') || it.contains('lucene-core') || it.contains('lucene-queries') || it.contains('lucene-queryparser') || it.contains('lucene-analyzers-phonetic') || + // dependencies from geode-protobuf it.contains('protobuf-java') } diff --git a/geode-assembly/src/test/java/org/apache/geode/management/internal/cli/commands/StatusLocatorRealGfshTest.java b/geode-assembly/src/test/java/org/apache/geode/management/internal/cli/commands/StatusLocatorRealGfshTest.java index 94ddc8f322..7e7b2bdca8 100644 --- a/geode-assembly/src/test/java/org/apache/geode/management/internal/cli/commands/StatusLocatorRealGfshTest.java +++ b/geode-assembly/src/test/java/org/apache/geode/management/internal/cli/commands/StatusLocatorRealGfshTest.java

@@ -30,6 +30,7 @@ @Test public void statusLocatorSucceedsWhenConnected() throws Exception { GfshScript.of("start locator --name=locator1").execute(gfshRule); + GfshScript.of("connect", "status locator --name=locator1").execute(gfshRule); } diff --git a/geode-core/src/main/java/org/apache/geode/distributed/ConfigurationProperties.java b/geode-core/src/main/java/org/apache/geode/distributed/ConfigurationProperties.java index 4944e984bd..649bfad407 100644 --- a/geode-core/src/main/java/org/apache/geode/distributed/ConfigurationProperties.java +++ b/geode-core/src/main/java/org/apache/geode/distributed/ConfigurationProperties.java @@ -1860,10 +1860,7 @@ * <U>Default</U>: <code>""</code> * </p> * <U>Since</U>: GemFire 8.0 - * - * @deprecated Since Geode 1.4 (See GEODE-3007) */ - @Deprecated() String USER_COMMAND_PACKAGES = "user-command-packages"; /** * The static String definition of the "off-heap-memory-size" property <a diff --git a/geode-core/src/main/java/org/apache/geode/distributed/internal/AbstractDistributionConfig.java b/geode-core/src/main/java/org/apache/geode/distributed/internal/AbstractDistributionConfig.java index 7c5929c4a8..43c5cb48fe 100644 --- a/geode-core/src/main/java/org/apache/geode/distributed/internal/AbstractDistributionConfig.java +++ b/geode-core/src/main/java/org/apache/geode/distributed/internal/AbstractDistributionConfig.java @@ -1018,6 +1018,9 @@ public static Class _getAttributeType(String attName) { m.put(GROUPS, "A comma separated list of all the groups this member belongs to." + " Defaults to \"\"."); + m.put(USER_COMMAND_PACKAGES, + "A comma separated list of the names of the packages containing classes that implement user commands."); + m.put(JMX_MANAGER, "If true then this member is willing to be a jmx manager. Defaults to false except on a locator."); m.put(JMX_MANAGER_START, diff --git a/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfig.java b/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfig.java index 3aeb19fad2..eabe6ccee6 100644 --- a/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfig.java +++ b/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfig.java @@ -386,6 +386,34 @@ */ File DEFAULT_DEPLOY_WORKING_DIR = new File(System.getProperty("user.dir")); + /** + * Returns the value of the {@link ConfigurationProperties#USER_COMMAND_PACKAGES} property + */ + @ConfigAttributeGetter(name = USER_COMMAND_PACKAGES) + String getUserCommandPackages(); + + /** + * Sets the system's user command path. + * + * @throws IllegalArgumentException if the specified value is not acceptable. + * @throws org.apache.geode.UnmodifiableException if this attribute can not be modified. + * @throws org.apache.geode.GemFireIOException if the set failure is caused by an error when + * writing to the system's configuration file. + */ + @ConfigAttributeSetter(name = USER_COMMAND_PACKAGES) + void setUserCommandPackages(String value); + + /** + * The name of the {@link ConfigurationProperties#USER_COMMAND_PACKAGES} property. + */ + @ConfigAttribute(type = String.class) + String USER_COMMAND_PACKAGES_NAME = USER_COMMAND_PACKAGES; + + /** + * The default value of the {@link ConfigurationProperties#USER_COMMAND_PACKAGES} property + */ + String DEFAULT_USER_COMMAND_PACKAGES = ""; + /** * Returns the value of the {@link ConfigurationProperties#LOG_FILE} property * diff --git a/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfigImpl.java b/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfigImpl.java index 0c13480702..cceeaf9eb0 100644 --- a/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfigImpl.java +++ b/geode-core/src/main/java/org/apache/geode/distributed/internal/DistributionConfigImpl.java @@ -625,6 +625,8 @@ private Map<String, ConfigSource> sourceMap = Collections.synchronizedMap(new HashMap<String, ConfigSource>()); + protected String userCommandPackages = DEFAULT_USER_COMMAND_PACKAGES; + /** * "off-heap-memory-size" with value of "" or "<size> [g|m]" */ @@ -757,6 +759,7 @@ public DistributionConfigImpl(DistributionConfig other) { this.redisPort = other.getRedisPort(); this.redisBindAddress = other.getRedisBindAddress(); this.redisPassword = other.getRedisPassword(); + this.userCommandPackages = other.getUserCommandPackages(); // following added for 8.0 this.enableSharedConfiguration = other.getEnableClusterConfiguration(); @@ -1831,6 +1834,10 @@ public int getAsyncMaxQueueSize() { return this.asyncMaxQueueSize; } + public String getUserCommandPackages() { + return this.userCommandPackages; + } + public int getHttpServicePort() { return this.httpServicePort; } @@ -1855,6 +1862,10 @@ public void setStartDevRestApi(boolean value) { this.startDevRestApi = value; } + public void setUserCommandPackages(String value) { + this.userCommandPackages = value; + } + public boolean getDeltaPropagation() { return this.deltaPropagation; } @@ -3002,8 +3013,9 @@ public boolean equals(final Object o) { .append(sslTrustStore, that.sslTrustStore) .append(sslTrustStorePassword, that.sslTrustStorePassword) .append(locatorSSLAlias, that.locatorSSLAlias).append(sslDefaultAlias, that.sslDefaultAlias) - .append(sourceMap, that.sourceMap).append(offHeapMemorySize, that.offHeapMemorySize) -

.append(shiroInit, that.shiroInit).isEquals(); + .append(sourceMap,
that.sourceMap).append(userCommandPackages, that.userCommandPackages) +
.append(offHeapMemorySize, that.offHeapMemorySize).append(shiroInit, that.shiroInit) + .isEquals(); } /**
@@ -3071,8 +3083,9 @@ public int hashCode() {
.append(sslCiphers).append(sslRequireAuthentication).append(sslKeyStore)
.append(sslKeyStoreType).append(sslKeyStorePassword).append(sslTrustStore)
.append(sslTrustStorePassword).append(sslWebServiceRequireAuthentication) -
.append(locatorSSLAlias).append(sslDefaultAlias).append(sourceMap).append(offHeapMemorySize) -
.append(lockMemory).append(shiroInit).append(modifiable).toHashCode(); +
.append(locatorSSLAlias).append(sslDefaultAlias).append(sourceMap) +
.append(userCommandPackages).append(offHeapMemorySize).append(lockMemory).append(shiroInit) +
.append(modifiable).toHashCode(); } /** diff --git a/geode-
core/src/main/java/org/apache/geode/management/internal/beans/MemberMBeanBridge.java b/geode-
core/src/main/java/org/apache/geode/management/internal/beans/MemberMBeanBridge.java index
f84799f40f..2a6565ace2 100644 --- a/geode-
core/src/main/java/org/apache/geode/management/internal/beans/MemberMBeanBridge.java +++ b/geode-
core/src/main/java/org/apache/geode/management/internal/beans/MemberMBeanBridge.java @@ -338,7
+338,8 @@ public MemberMBeanBridge(InternalCache cache, SystemManagementService service) {
this.config = system.getConfig(); try { - this.commandProcessor = new
OnlineCommandProcessor(cache.getSecurityService()); + this.commandProcessor = + new
OnlineCommandProcessor(system.getProperties(), cache.getSecurityService()); } catch (Exception e) {
commandServiceInitError = e.getMessage(); logger.info(LogMarker.CONFIG, "Command processor could
not be initialized. {}", diff --git a/geode-
core/src/main/java/org/apache/geode/management/internal/cli/CommandManager.java b/geode-
core/src/main/java/org/apache/geode/management/internal/cli/CommandManager.java index
0b13d62cf4..44617d3f18 100644 --- a/geode-
core/src/main/java/org/apache/geode/management/internal/cli/CommandManager.java +++ b/geode-
core/src/main/java/org/apache/geode/management/internal/cli/CommandManager.java @@ -14,93 +14,293
@@ */ package org.apache.geode.management.internal.cli; +import static
org.apache.geode.distributed.ConfigurationProperties.USER_COMMAND_PACKAGES; -import static
java.util.stream.Collectors.toSet; - -import java.util.Objects; +import java.lang.reflect.Method; +import
java.util.ArrayList; +import java.util.HashSet; +import java.util.Iterator; +import java.util.List; +import
java.util.Properties; +import java.util.ServiceConfigurationError; +import java.util.ServiceLoader; import
java.util.Set; -import java.util.stream.Stream; +import java.util.StringTokenizer; +import
org.springframework.shell.converters.EnumConverter; +import
org.springframework.shell.converters.SimpleFileConverter; import
org.springframework.shell.core.CommandMarker; import org.springframework.shell.core.Converter; +import
org.springframework.shell.core.MethodTarget; +import
org.springframework.shell.core.annotation.CliAvailabilityIndicator; +import
org.springframework.shell.core.annotation.CliCommand; +import
org.apache.geode.distributed.ConfigurationProperties; +import
org.apache.geode.distributed.internal.DistributionConfig; +import
org.apache.geode.internal.ClassPathLoader; +import
org.apache.geode.management.internal.cli.commands.GfshCommand; import
org.apache.geode.management.internal.cli.help.Helper; import
org.apache.geode.management.internal.cli.shell.Gfsh; import
org.apache.geode.management.internal.cli.util.ClasspathScanLoadHelper; /** + * * this only takes care of
loading all available command markers and converters from the application * * @since GemFire 7.0 */ public
class CommandManager { - // Skip some of the Converters from Spring Shell for our customization - private
static final Set<String> EXCLUDED_CLASSES = - Stream.of("-
org.springframework.shell.converters.SimpleFileConverter", - "-
org.springframework.shell.converters.FileConverter", - "-
org.springframework.shell.converters.EnumConverter", - "-
org.springframework.shell.commands.ExitCommands", - "-
org.springframework.shell.commands.HelpCommands", - "-
org.springframework.shell.commands.VersionCommands", - "-
org.springframework.shell.commands.ConsoleCommands").collect(toSet()); - - private final LogWrapper
logWrapper = LogWrapper.getInstance(); - private final Set<CommandMarker> commandMarkers; - private
final Set<Converter> converters; - private final Helper helper; + public static final String
USER_CMD_PACKAGES_PROPERTY = + DistributionConfig.GEMFIRE_PREFIX +
USER_COMMAND_PACKAGES; + public static final String

USER_CMD_PACKAGES_ENV_VARIABLE = "GEMFIRE_USER_COMMAND_PACKAGES"; + private static final Object INSTANCE_LOCK = new Object(); + + private final Helper helper = new Helper(); + + private final List<Converter<?>> converters = new ArrayList<Converter<?>>(); + private final List<CommandMarker> commandMarkers = new ArrayList<>(); + private Properties cacheProperties; + private LogWrapper logWrapper; + + /** + * this constructor is used from Gfsh VM. We are getting the user-command-package from system + * environment. used by Gfsh. + */ public CommandManager() { - helper = new Helper(); - converters = loadConverters(); - commandMarkers = loadCommandMarkers(); + this(null); } - private Set<Converter> loadConverters() { - Set<Converter> converters = instantiateAllClassesImplementing(Converter.class); - raiseExceptionIfEmpty(converters, "converters"); + /** + * this is used when getting the instance in a cache server. We are getting the + * user-command-package from distribution properties. used by OnlineCommandProcessor. + */ + public CommandManager(final Properties cacheProperties) { + if (cacheProperties != null) { + this.cacheProperties = cacheProperties; + } + logWrapper = LogWrapper.getInstance(); + loadCommands(); + } - converters.forEach(this::setContextIfCommandManagerAware); - return converters; + private static void raiseExceptionIfEmpty(Set<Class<?>> foundClasses, String errorFor) + throws IllegalStateException { + if (foundClasses == null || foundClasses.isEmpty()) { + throw new IllegalStateException(+ "Required " + errorFor + " classes were not loaded. Check logs for errors."); + } } - private Set<CommandMarker> loadCommandMarkers() { - Set<CommandMarker> commandMarkers = instantiateAllClassesImplementing(CommandMarker.class); - raiseExceptionIfEmpty(commandMarkers, "commandMarkers"); + private void loadUserCommands() { + final Set<String> userCommandPackages = new HashSet<String>(); - commandMarkers.forEach(this::setContextIfCommandManagerAware); - commandMarkers.forEach(helper::registerCommand); - return commandMarkers; + // Find by packages specified by the system property + if (System.getProperty(USER_CMD_PACKAGES_PROPERTY) != null) { + StringTokenizer tokenizer = new StringTokenizer(System.getProperty(USER_CMD_PACKAGES_PROPERTY), ","); + while (tokenizer.hasMoreTokens()) { + userCommandPackages.add(tokenizer.nextToken()); + } + } + + // Find by packages specified by the environment variable + if (System.getenv().containsKey(USER_CMD_PACKAGES_ENV_VARIABLE)) { + StringTokenizer tokenizer = new StringTokenizer(System.getenv().get(USER_CMD_PACKAGES_ENV_VARIABLE), ","); + while (tokenizer.hasMoreTokens()) { + userCommandPackages.add(tokenizer.nextToken()); + } + } + + // Find by packages specified in the distribution config + if (this.cacheProperties != null) { + String cacheUserCmdPackages = this.cacheProperties.getProperty(ConfigurationProperties.USER_COMMAND_PACKAGES); + if (cacheUserCmdPackages != null && !cacheUserCmdPackages.isEmpty()) { + StringTokenizer tokenizer = new StringTokenizer(cacheUserCmdPackages, ","); + while (tokenizer.hasMoreTokens()) { + userCommandPackages.add(tokenizer.nextToken()); + } + } + } + + // Load commands found in all of the packages + for (String userCommandPackage : userCommandPackages) { + try { + Set<Class<?>> foundClasses = ClasspathScanLoadHelper + .scanPackageForClassesImplementing(userCommandPackage, CommandMarker.class); + for (Class<?> klass : foundClasses) { + try { + add((CommandMarker) klass.newInstance()); + } catch (Exception e) { + logWrapper.warning("Could not load User Commands from: " + klass + " due to " + + e.getLocalizedMessage()); // continue + } + } + raiseExceptionIfEmpty(foundClasses, "User Command"); + } catch (IllegalStateException e) { + logWrapper.warning(e.getMessage(), e); + throw e; + } + } + } + + /** + * Loads commands via {@link ServiceLoader} from {@link ClassPathLoader}. + * + * @since GemFire 8.1 + */ + private void loadPluginCommands() { + final Iterator<CommandMarker> iterator = ServiceLoader + .load(CommandMarker.class, ClassPathLoader.getLatest().asClassLoader()).iterator(); + while (iterator.hasNext()) { + try { + final CommandMarker commandMarker = iterator.next(); + try { + add(commandMarker); + } catch (Exception e) { + logWrapper.warning("Could not load Command from: " + commandMarker.getClass() + " due to " + + e.getLocalizedMessage(), e); // continue + } + } catch (ServiceConfigurationError e) { + logWrapper.severe("Could not load Command: " + e.getLocalizedMessage(), e); // continue + } + } } - private <T> Set<T> instantiateAllClassesImplementing(Class<T> implementedInterface) { - Set<Class<? extends T>> classes = ClasspathScanLoadHelper.scanClasspathForClassesImplementing(- implementedInterface, EXCLUDED_CLASSES.toArray(new String[0])); - return classes.stream().map(this::instantiateClass).filter(Objects::nonNull).collect(toSet()); + private void loadCommands() { + loadUserCommands(); + loadPluginCommands(); + loadGeodeCommands(); + loadConverters(); } - private <T> T instantiateClass(Class<T> classToInstantiate) { + private void loadConverters() { + Set<Class<?>> foundClasses; + // Converters try { - return classToInstantiate.newInstance(); - } catch (Exception e) { - logWrapper.warning("Could not load command or converter from: " + classToInstantiate, e); + foundClasses = ClasspathScanLoadHelper.scanPackageForClassesImplementing(

"org.apache.geode.management.internal.cli.converters", Converter.class); + for (Class<?> klass : foundClasses) { + try { + Converter<?> object = (Converter<?>) klass.newInstance(); + add(object); + + } catch (Exception e) { + logWrapper.warning(+ "Could not load Converter from: " + klass + " due to " + e.getLocalizedMessage()); // continue + } + } + raiseExceptionIfEmpty(foundClasses, "Converters"); + + // Spring shell's converters + foundClasses = ClasspathScanLoadHelper.scanPackageForClassesImplementing(+ "org.springframework.shell.converters", Converter.class); + for (Class<?> klass : foundClasses) { + if (!SHL_CONVERTERS_TOSKIP.contains(klass)) { + try { + add((Converter<?>) klass.newInstance()); + } catch (Exception e) { + logWrapper.warning(+ "Could not load Converter from: " + klass + " due to " + e.getLocalizedMessage()); // continue + } + } + } + raiseExceptionIfEmpty(foundClasses, "Basic Converters"); + } catch (IllegalStateException e) { + logWrapper.warning(e.getMessage(), e); + throw e; } - return null; } - private void setContextIfCommandManagerAware(Object commandOrConverter) { - if (CommandManagerAware.class.isAssignableFrom(commandOrConverter.getClass())) { - ((CommandManagerAware) commandOrConverter).setCommandManager(this); + private void loadGeodeCommands() { + // CommandMarkers + Set<Class<?>> foundClasses; + try { + // geode's commands + foundClasses = ClasspathScanLoadHelper.scanPackageForClassesImplementing(+ GfshCommand.class.getPackage().getName(), CommandMarker.class); + + for (Class<?> klass : foundClasses) { + try { + add((CommandMarker) klass.newInstance()); + } catch (Exception e) { + logWrapper.warning(+ "Could not load Command from: " + klass + " due to " + e.getLocalizedMessage()); // continue + } + } + raiseExceptionIfEmpty(foundClasses, "Commands"); + + // do not add Spring shell's commands for now. When we add it, we need to tell the parser that + // these are offline commands. + } catch (IllegalStateException e) { + logWrapper.warning(e.getMessage(), e); + throw e; } } - private static void raiseExceptionIfEmpty(Set<?> foundClasses, String classType) - throws IllegalStateException { - if (foundClasses == null || foundClasses.isEmpty()) { - throw new IllegalStateException("No " + classType + " were loaded. Check logs for errors."); + /** Skip some of the Converters from Spring Shell for our customization */ + private static List<Class> SHL_CONVERTERS_TOSKIP = new ArrayList(); + static { + // skip springs SimpleFileConverter to use our own FilePathConverter + SHL_CONVERTERS_TOSKIP.add(SimpleFileConverter.class); + // skip spring's EnumConverter to use our own EnumConverter + SHL_CONVERTERS_TOSKIP.add(EnumConverter.class); + } + + public List<Converter<?>> getConverters() { + return converters; + } + + public List<CommandMarker> getCommandMarkers() { + return commandMarkers; + } + + /** + * Method to add new Converter + * + * @param converter + */ + void add(Converter<?> converter) { + if (CommandManagerAware.class.isAssignableFrom(converter.getClass())) { + ((CommandManagerAware) converter).setCommandManager(this); + } + converters.add(converter); + } + + /** + * Method to add new Commands to the parser + * + * @param commandMarker + */ + void add(CommandMarker commandMarker) { + if (CommandManagerAware.class.isAssignableFrom(commandMarker.getClass())) { + ((CommandManagerAware) commandMarker).setCommandManager(this); + } + commandMarkers.add(commandMarker); + for (Method method : commandMarker.getClass().getMethods()) { + CliCommand cliCommand = method.getAnnotation(CliCommand.class); + CliAvailabilityIndicator availability = method.getAnnotation(CliAvailabilityIndicator.class); + if (cliCommand == null && availability == null) { + continue; + } + + if (cliCommand != null) { + helper.addCommand(cliCommand, method); + } + + if (availability != null) { + helper.addAvailabilityIndicator(availability, new MethodTarget(method, commandMarker)); + } } } + public Helper getHelper() { + return helper; + } + public String obtainHelp(String buffer) { int terminalWidth = -1; Gfsh gfsh = Gfsh.getCurrentInstance(); @@ -114,15 +314,4 @@ public String obtainHint(String topic) { return helper.getHint(topic); } - public Set<Converter> getConverters() { - return converters; - } - - public Set<CommandMarker> getCommandMarkers() { - return commandMarkers; - } - - public Helper getHelper() { - return helper; - } } diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/GfshParser.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/GfshParser.java index 4d3b546f1e..4fc7d4e8b0 100755 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/GfshParser.java +++ b/geode-core/src/main/java/org/apache/geode/management/internal/cli/GfshParser.java @@ -57,14 +57,10 @@ public GfshParser(CommandManager commandManager) { add(command); } - for (Converter converter : commandManager.getConverters()) { + for (Converter<?> converter : commandManager.getConverters()) { if (converter.getClass().isAssignableFrom(ArrayConverter.class)) { ArrayConverter arrayConverter = (ArrayConverter) converter; - HashSet<Converter<?>> converters = new HashSet<>(); - for (Converter c : commandManager.getConverters()) { - converters.add(c); - } - arrayConverter.setConverters(converters); + arrayConverter.setConverters(new HashSet<>(commandManager.getConverters())); } add(converter); } diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/help/Helper.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/help/Helper.java index ec6f4e3ae1..917f3e4961 100644 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/help/Helper.java +++ b/geode-

core/src/main/java/org/apache/geode/management/internal/cli/help/Helper.java @@ -14,10 +14,9 @@ */ package org.apache.geode.management.internal.cli.help; -import static java.util.stream.Collectors.joining; - import java.lang.annotation.Annotation; import java.lang.reflect.Method; +import java.util.ArrayList; import java.util.Arrays; import java.util.Collection; import java.util.Collections; @@ -28,7 +27,6 @@ import java.util.TreeMap; import org.apache.commons.lang.StringUtils; -import org.springframework.shell.core.CommandMarker; import org.springframework.shell.core.MethodTarget; import org.springframework.shell.core.annotation.CliAvailabilityIndicator; import org.springframework.shell.core.annotation.CliCommand; @@ -67,7 +65,8 @@ private final Map<String, Topic> topics = new HashMap<>(); private final Map<String, Method> commands = new TreeMap<String, Method>(); - private final Map<String, MethodTarget> availabilityIndicators = new HashMap<>(); + private final Map<String, MethodTarget> availabilityIndicators = + new HashMap<String, MethodTarget>(); public Helper() { initTopic(CliStrings.DEFAULT_TOPIC_GEODE, CliStrings.DEFAULT_TOPIC_GEODE__DESC); @@ -95,25 +94,7 @@ private void initTopic(String topic, String desc) { topics.put(topic, new Topic(topic, desc)); } - public void registerCommand(CommandMarker commandMarker) { - for (Method method : commandMarker.getClass().getMethods()) { - CliCommand cliCommand = method.getAnnotation(CliCommand.class); - CliAvailabilityIndicator availability = method.getAnnotation(CliAvailabilityIndicator.class); - if (cliCommand == null && availability == null) { - continue; - } - if (cliCommand != null) { - addCommand(cliCommand, method); - } - if (availability != null) { - addAvailabilityIndicator(availability, new MethodTarget(method, commandMarker)); - } - } - } - - protected void addCommand(CliCommand command, Method commandMethod) { + public void addCommand(CliCommand command, Method commandMethod) { // put all the command synonyms in the command map Arrays.stream(command.value()).forEach(cmd -> { commands.put(cmd, commandMethod); @@ -139,8 +120,7 @@ protected void addCommand(CliCommand command, Method commandMethod) { }); } - private void addAvailabilityIndicator(CliAvailabilityIndicator availability, - MethodTarget target) { + public void addAvailabilityIndicator(CliAvailabilityIndicator availability, MethodTarget target) { Arrays.stream(availability.value()).forEach(command -> { availabilityIndicators.put(command, target); }); @@ -172,7 +152,9 @@ public String getHint(String buffer) { builder.append(CliStrings.HINT__MSG__TOPICS_AVAILABLE).append(GfshParser.LINE_SEPARATOR) .append(GfshParser.LINE_SEPARATOR); - topics.keySet().stream().sorted() + List<String> sortedTopics = new ArrayList<>(topics.keySet()); + Collections.sort(sortedTopics); + sortedTopics.stream() .forEachOrdered(topic -> builder.append(topic).append(GfshParser.LINE_SEPARATOR)); return builder.toString(); } @@ -183,7 +165,8 @@ public String getHint(String buffer) { } builder.append(topic.desc).append(GfshParser.LINE_SEPARATOR).append(GfshParser.LINE_SEPARATOR); - topic.relatedCommands.stream().sorted().forEach(command -> builder.append(command.command) + Collections.sort(topic.relatedCommands); + topic.relatedCommands.stream().forEachOrdered(command -> builder.append(command.command) .append(": ").append(command.desc).append(GfshParser.LINE_SEPARATOR)); return builder.toString(); } @@ -192,7 +175,7 @@ public String getHint(String buffer) { return topics.keySet(); } - private boolean isAvailable(String command) { + boolean isAvailable(String command) { MethodTarget target = availabilityIndicators.get(command); if (target == null) { return true; @@ -278,12 +261,16 @@ HelpBlock getOptionDetail(CliOption cliOption) { HelpBlock optionNode = new HelpBlock(getPrimaryKey(cliOption)); String help = cliOption.help(); optionNode.addChild(new HelpBlock((StringUtils.isNotBlank(help) ? help : ""))); - String synonyms = getSynonyms(cliOption).stream().collect(joining(",")); - if (StringUtils.isNotEmpty(synonyms)) { - optionNode.addChild(new HelpBlock(SYNONYMS_SUB_NAME + synonyms)); + if (getSynonyms(cliOption).size() > 0) { + StringBuilder builder = new StringBuilder(); + for (String string : getSynonyms(cliOption)) { + if (builder.length() > 0) { + builder.append(","); + } + builder.append(string); + } + optionNode.addChild(new HelpBlock(SYNONYMS_SUB_NAME + builder.toString())); } optionNode.addChild(new HelpBlock(REQUIRED_SUB_NAME + ((cliOption.mandatory()) ? TRUE_TOKEN : FALSE_TOKEN))); if (!isNullOrBlank(cliOption.specifiedDefaultValue())) { @@ -372,14 +359,18 @@ private static String getPrimaryKey(CliOption option) { } private static List<String> getSynonyms(CliOption option) { - List<String> synonyms = new ArrayList<>(); String[] keys = option.key(); if (keys.length < 2) - return Collections.emptyList(); + return synonyms; // if the primary key is empty (like sh and help command), then there should be no synonyms. if ("".equals(keys[0])) - return Collections.emptyList(); - return Arrays.asList(keys).subList(1, keys.length); + return synonyms; + for (int i = 1; i < keys.length; i++) { + synonyms.add(keys[i]); + } + return synonyms; } private static boolean isNullOrBlank(String value) { diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/MemberCommandService.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/MemberCommandService.java index a4c6f59802..6130117d2e 100644 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/MemberCommandService.java +++

b/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/MemberCommandService.java @@ -35,7 +35,8 @@ public MemberCommandService(InternalCache cache) throws CommandServiceException { this.cache = cache; try { - this.onlineCommandProcessor = new OnlineCommandProcessor(cache.getSecurityService()); + this.onlineCommandProcessor = new OnlineCommandProcessor(+ cache.getDistributedSystem().getProperties(), cache.getSecurityService()); } catch (Exception e) { throw new CommandServiceException("Could not load commands.", e); } diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessor.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessor.java index 38ab364392..7fa4acba51 100644 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessor.java +++ b/geode-core/src/main/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessor.java @@ -18,6 +18,7 @@ import java.lang.reflect.Method; import java.util.Collections; import java.util.Map; +import java.util.Properties; import org.springframework.shell.core.Parser; import org.springframework.shell.event.ParseResult; @@ -25,6 +26,7 @@ import org.apache.geode.annotations.TestingOnly; import org.apache.geode.internal.security.SecurityService; +import org.apache.geode.internal.security.SecurityServiceFactory; import org.apache.geode.management.cli.CliMetaData; import org.apache.geode.management.cli.CommandProcessingException; import org.apache.geode.management.cli.Result; @@ -47,14 +49,20 @@ private final SecurityService securityService; - public OnlineCommandProcessor(SecurityService securityService) + @TestingOnly + public OnlineCommandProcessor() throws ClassNotFoundException, IOException { + this(new Properties(), SecurityServiceFactory.create()); + } + + public OnlineCommandProcessor(Properties cacheProperties, SecurityService securityService) throws ClassNotFoundException, IOException { - this(securityService, new CommandExecutor()); + this(cacheProperties, securityService, new CommandExecutor()); } @TestingOnly - public OnlineCommandProcessor(SecurityService securityService, CommandExecutor commandExecutor) { - this.gfshParser = new GfshParser(new CommandManager()); + public OnlineCommandProcessor(Properties cacheProperties, SecurityService securityService, + CommandExecutor commandExecutor) { + this.gfshParser = new GfshParser(new CommandManager(cacheProperties)); this.executor = commandExecutor; this.securityService = securityService; } diff --git a/geode-core/src/main/java/org/apache/geode/management/internal/cli/util/ClasspathScanLoadHelper.java b/geode-core/src/main/java/org/apache/geode/management/internal/cli/util/ClasspathScanLoadHelper.java index 11da4b208a..046d890567 100644 --- a/geode-core/src/main/java/org/apache/geode/management/internal/cli/util/ClasspathScanLoadHelper.java +++ b/geode-core/src/main/java/org/apache/geode/management/internal/cli/util/ClasspathScanLoadHelper.java @@ -21,7 +21,6 @@ import java.util.Set; import io.github.lukehutch.fastclasspathscanner.FastClasspathScanner; -import io.github.lukehutch.fastclasspathscanner.matchprocessor.ImplementingClassMatchProcessor; /** * Utility class to scan class-path & load classes. @@ -29,26 +28,18 @@ * @since GemFire 7.0 */ public class ClasspathScanLoadHelper { - - public static <T> Set<Class<? extends T>> scanClasspathForClassesImplementing(- Class<T> implementedInterface, String... packageSpec) { - return scanPackageForClassesImplementing(implementedInterface, packageSpec); - } - - public static <T> Set<Class<? extends T>> scanPackageForClassesImplementing(- Class<T> implementedInterface, String... packageSpec) { - Set<Class<? extends T>> classesImplementing = new HashSet<>(); - ImplementingClassMatchProcessor<T> matchProcessor = classesImplementing::add; - new FastClasspathScanner(packageSpec) - .matchClassesImplementing(implementedInterface, matchProcessor).scan(); + public static Set<Class<?>> scanPackageForClassesImplementing(String packageToScan, + Class<?> implementedInterface) { + Set<Class<?>> classesImplementing = new HashSet<>(); + new FastClasspathScanner(packageToScan) + .matchClassesImplementing(implementedInterface, classesImplementing::add).scan(); return classesImplementing.stream().filter(ClasspathScanLoadHelper::isInstantiable) .collect(toSet()); } - private static <T> boolean isInstantiable(Class<T> classToInstantiate) { - int modifiers = classToInstantiate.getModifiers(); + private static boolean isInstantiable(Class<?> klass) { + int modifiers = klass.getModifiers(); return !Modifier.isAbstract(modifiers) && !Modifier.isInterface(modifiers) && Modifier.isPublic(modifiers); diff --git a/geode-core/src/test/java/org/apache/geode/distributed/internal/DistributionConfigJUnitTest.java b/geode-core/src/test/java/org/apache/geode/distributed/internal/DistributionConfigJUnitTest.java index e874898d82..7e5f83f4f2 100644 --- a/geode-

core/src/test/java/org/apache/geode/distributed/internal/DistributionConfigJUnitTest.java +++ b/geode-core/src/test/java/org/apache/geode/distributed/internal/DistributionConfigJUnitTest.java @@ -99,7 +99,7 @@ public void before() { @Test public void testGetAttributeNames() { String[] attNames = AbstractDistributionConfig._getAttNames(); - assertEquals(156, attNames.length); + assertEquals(attNames.length, 157); List boolList = new ArrayList(); List intList = new ArrayList(); @@ -135,7 +135,7 @@ public void testGetAttributeNames() { // are. assertEquals(29, boolList.size()); assertEquals(33, intList.size()); - assertEquals(85, stringList.size()); + assertEquals(86, stringList.size()); assertEquals(5, fileList.size()); assertEquals(4, otherList.size()); } diff --git a/geode-core/src/test/java/org/apache/geode/management/internal/cli/ClasspathScanLoadHelperJUnitTest.java b/geode-core/src/test/java/org/apache/geode/management/internal/cli/ClasspathScanLoadHelperJUnitTest.java index e48c8d24a7..1dfd61d9a1 100644 --- a/geode-core/src/test/java/org/apache/geode/management/internal/cli/ClasspathScanLoadHelperJUnitTest.java +++ b/geode-core/src/test/java/org/apache/geode/management/internal/cli/ClasspathScanLoadHelperJUnitTest.java @@ -14,8 +14,7 @@ */ package org.apache.geode.management.internal.cli; -import static org.junit.Assert.assertEquals; -import static org.junit.Assert.assertTrue; +import static org.junit.Assert.*; import java.util.Set; @@ -36,36 +35,36 @@ private final String PACKAGE_NAME = "org.apache.geode.management.internal.cli.domain"; private final String WRONG_PACKAGE_NAME = "org.apache.geode.management.internal.cli.domain1"; - private final Class INTERFACE1 = Interface1.class; - private final Class NO_IMPL_INTERFACE = Versionable.class; - private final Class INTERFACE2 = Interface2.class; - private final Class IMPL1 = Impl1.class; - private final Class IMPL2 = Impl12.class; - private final Class ABSTRACT_IMPL = AbstractImpl.class; + private final Class<?> INTERFACE1 = Interface1.class; + private final Class<?> NO_IMPL_INTERFACE = Versionable.class; + private final Class<?> INTERFACE2 = Interface2.class; + private final Class<?> IMPL1 = Impl1.class; + private final Class<?> IMPL2 = Impl12.class; + private final Class<?> ABSTRACT_IMPL = AbstractImpl.class; @Test public void testLoadAndGet() throws Exception { Set<Class<?>> classLoaded = - ClasspathScanLoadHelper.scanPackageForClassesImplementing(INTERFACE1, PACKAGE_NAME); + ClasspathScanLoadHelper.scanPackageForClassesImplementing(PACKAGE_NAME, INTERFACE1); assertEquals(2, classLoaded.size()); assertTrue(classLoaded.contains(IMPL1)); assertTrue(classLoaded.contains(IMPL2)); classLoaded = - ClasspathScanLoadHelper.scanPackageForClassesImplementing(INTERFACE2, PACKAGE_NAME); + ClasspathScanLoadHelper.scanPackageForClassesImplementing(PACKAGE_NAME, INTERFACE2); assertEquals(1, classLoaded.size()); assertTrue(classLoaded.contains(IMPL2)); classLoaded = - ClasspathScanLoadHelper.scanPackageForClassesImplementing(INTERFACE2, WRONG_PACKAGE_NAME); + ClasspathScanLoadHelper.scanPackageForClassesImplementing(WRONG_PACKAGE_NAME, INTERFACE2); assertEquals(0, classLoaded.size()); classLoaded = - ClasspathScanLoadHelper.scanPackageForClassesImplementing(NO_IMPL_INTERFACE, PACKAGE_NAME); + ClasspathScanLoadHelper.scanPackageForClassesImplementing(PACKAGE_NAME, NO_IMPL_INTERFACE); assertEquals(0, classLoaded.size()); - classLoaded = ClasspathScanLoadHelper.scanPackageForClassesImplementing(NO_IMPL_INTERFACE, - WRONG_PACKAGE_NAME); + classLoaded = ClasspathScanLoadHelper.scanPackageForClassesImplementing(WRONG_PACKAGE_NAME, + NO_IMPL_INTERFACE); assertEquals(0, classLoaded.size()); } } diff --git a/geode-core/src/test/java/org/apache/geode/management/internal/cli/CommandManagerJUnitTest.java b/geode-core/src/test/java/org/apache/geode/management/internal/cli/CommandManagerJUnitTest.java index c65ff04936..813347128c 100644 --- a/geode-core/src/test/java/org/apache/geode/management/internal/cli/CommandManagerJUnitTest.java +++ b/geode-core/src/test/java/org/apache/geode/management/internal/cli/CommandManagerJUnitTest.java @@ -18,12 +18,20 @@ import static org.junit.Assert.assertNotNull; import static org.junit.Assert.assertTrue; +import java.util.List; + import org.junit.Before; import org.junit.Test; import org.junit.experimental.categories.Category; import org.springframework.shell.core.CommandMarker; +import org.springframework.shell.core.Completion; +import org.springframework.shell.core.Converter; +import org.springframework.shell.core.MethodTarget; import org.springframework.shell.core.annotation.CliAvailabilityIndicator; import org.springframework.shell.core.annotation.CliCommand; +import org.springframework.shell.core.annotation.CliOption; +import org.apache.geode.management.cli.CliMetaData; import org.apache.geode.management.cli.Result; import org.apache.geode.management.internal.security.ResourceOperation; import

org.apache.geode.security.ResourcePermission.Operation; @@ -35,6 +43,49 @@ */ @Category(UnitTest.class) public class CommandManagerJUnitTest { + + private static final String COMMAND1_NAME = "command1"; + private static final String COMMAND1_NAME_ALIAS = "command1_alias"; + private static final String COMMAND2_NAME = "c2"; + + private static final String COMMAND1_HELP = "help for " + COMMAND1_NAME; + // ARGUMENTS + private static final String ARGUMENT1_NAME = "argument1"; + private static final String ARGUMENT1_HELP = "help for argument1"; + private static final String ARGUMENT1_CONTEXT = "context for argument 1"; + private static final Completion[] ARGUMENT1_COMPLETIONS = + {new Completion("arg1"), new Completion("arg1alt")}; + private static final String ARGUMENT2_NAME = "argument2"; + private static final String ARGUMENT2_CONTEXT = "context for argument 2"; + private static final String ARGUMENT2_HELP = "help for argument2"; + private static final String ARGUMENT2_UNSPECIFIED_DEFAULT_VALUE = + "{unspecified default value for argument2}"; + private static final Completion[] ARGUMENT2_COMPLETIONS = + {new Completion("arg2"), new Completion("arg2alt")}; + + // OPTIONS + private static final String OPTION1_NAME = "option1"; + private static final String OPTION1_SYNONYM = "opt1"; + private static final String OPTION1_HELP = "help for option1"; + private static final String OPTION1_CONTEXT = "context for option1"; + private static final String OPTION1_SPECIFIED_DEFAULT_VALUE = + "{specified default value for option1}"; + private static final Completion[] OPTION1_COMPLETIONS = + {new Completion("option1"), new Completion("option1Alternate")}; + private static final String OPTION2_NAME = "option2"; + private static final String OPTION2_HELP = "help for option2"; + private static final String OPTION2_CONTEXT = "context for option2"; + private static final String OPTION2_SPECIFIED_DEFAULT_VALUE = + "{specified default value for option2}"; + private static final String OPTION3_NAME = "option3"; + private static final String OPTION3_SYNONYM = "opt3"; + private static final String OPTION3_HELP = "help for option3"; + private static final String OPTION3_CONTEXT = "context for option3"; + private static final String OPTION3_SPECIFIED_DEFAULT_VALUE = + "{specified default value for option3}"; + private static final String OPTION3_UNSPECIFIED_DEFAULT_VALUE = + "{unspecified default value for option3}"; + private CommandManager commandManager; @Before @@ -60,6 +111,11 @@ public void testCommandManagerInstance() throws Exception { assertNotNull(commandManager); } + /** + * Tests {@link CommandManager#loadPluginCommands()}. + * + * @since GemFire 8.1 + */ @Test public void testCommandManagerLoadPluginCommands() throws Exception { assertNotNull(commandManager); @@ -67,10 +123,97 @@ public void testCommandManagerLoadPluginCommands() throws Exception { public void testCommandManagerLoadPluginCommands() throws Exception { // see META-INF/services/org.springframework.shell.core.CommandMarker service loader file. assertTrue("Should find listed plugin.", commandManager.getHelper().getCommands().contains("mock plugin command")); - assertThat(- commandManager.getCommandMarkers().stream().anyMatch(c -> c instanceof MockPluginCommand)); + assertTrue("Should not find unlisted plugin.", + !commandManager.getHelper().getCommands().contains("mock plugin command unlisted")); + } + + /** + * class that represents dummy commands + */ + public static class Commands implements CommandMarker { + + @CliCommand(value = {COMMAND1_NAME, COMMAND1_NAME_ALIAS}, help = COMMAND1_HELP) + @CliMetaData(shellOnly = true, relatedTopic = {"relatedTopicOfCommand1"}) + @ResourceOperation(resource = Resource.CLUSTER, operation = Operation.READ) + public static String command1(+ @CliOption(key = ARGUMENT1_NAME, optionContext = ARGUMENT1_CONTEXT, help = ARGUMENT1_HELP, + mandatory = true) String argument1, + @CliOption(key = ARGUMENT2_NAME, optionContext = ARGUMENT2_CONTEXT, help = ARGUMENT2_HELP, + unspecifiedDefaultValue = ARGUMENT2_UNSPECIFIED_DEFAULT_VALUE) String argument2, + @CliOption(key = {OPTION1_NAME, OPTION1_SYNONYM}, help = OPTION1_HELP, mandatory = true, + optionContext = OPTION1_CONTEXT, + specifiedDefaultValue = OPTION1_SPECIFIED_DEFAULT_VALUE) String option1, + @CliOption(key = {OPTION2_NAME}, help = OPTION2_HELP, optionContext = OPTION2_CONTEXT, + specifiedDefaultValue = OPTION2_SPECIFIED_DEFAULT_VALUE) String option2, + @CliOption(key = {OPTION3_NAME, OPTION3_SYNONYM}, help = OPTION3_HELP, + optionContext = OPTION3_CONTEXT, + unspecifiedDefaultValue = OPTION3_UNSPECIFIED_DEFAULT_VALUE, + specifiedDefaultValue = OPTION3_SPECIFIED_DEFAULT_VALUE) String option3) { + return null; + } + + @CliCommand(value = {COMMAND2_NAME}) + @ResourceOperation(resource = Resource.CLUSTER, operation = Operation.READ) + public static String command2() { + return null; + } + + @CliCommand(value = {"testParamConcat"}) + @ResourceOperation(resource = Resource.CLUSTER, operation = Operation.READ) + public static Result testParamConcat(@CliOption(key = {"string"}) String string, + @CliOption(key = {"stringArray"}) String[] stringArray, + @CliOption(key = {"integer"}) Integer integer, + @CliOption(key = {"colonArray"}) String[] colonArray) { + return null; + } + + @CliCommand(value = {"testMultiWordArg"}) + @ResourceOperation(resource = Resource.CLUSTER, operation = Operation.READ) + public static Result testMultiWordArg(@CliOption(key = "arg1") String arg1, + @CliOption(key = "arg2") String arg2) { + return null; + } + +

@CliAvailabilityIndicator({COMMAND1_NAME}) + public boolean isAvailable() { + return true; // always available on server + } } + /** + * Used by testCommandManagerLoadPluginCommands + */ + private static class SimpleConverter implements Converter<String> { + + @Override + public boolean supports(Class<?> type, String optionContext) { + return type.isAssignableFrom(String.class); + } + + @Override + public String convertFromText(String value, Class<?> targetType, String optionContext) { + return value; + } + + @Override + public boolean getAllPossibleValues(List<Completion> completions, Class<?> targetType, + String existingData, String context, MethodTarget target) { + if (context.equals(ARGUMENT1_CONTEXT)) { + for (Completion completion : ARGUMENT1_COMPLETIONS) { + completions.add(completion); + } + } else if (context.equals(ARGUMENT2_CONTEXT)) { + for (Completion completion : ARGUMENT2_COMPLETIONS) { + completions.add(completion); + } + } else if (context.equals(OPTION1_CONTEXT)) { + for (Completion completion : OPTION1_COMPLETIONS) { + completions.add(completion); + } + } + return true; + } + } public static class MockPluginCommand implements CommandMarker { @CliCommand(value = "mock plugin command") @@ -79,4 +222,13 @@ public Result mockPluginCommand() { return null; } } + + public static class MockPluginCommandUnlisted implements CommandMarker { + @CliCommand(value = "mock plugin command unlisted") + @ResourceOperation(resource = Resource.CLUSTER, operation = Operation.READ) + public Result mockPluginCommandUnlisted() { + return null; + } + } + } diff --git a/geode-core/src/test/java/org/apache/geode/management/internal/cli/commands/CliCommandTestBase.java b/geode-core/src/test/java/org/apache/geode/management/internal/cli/commands/CliCommandTestBase.java index d79993201b..5714829cfa 100644 --- a/geode-core/src/test/java/org/apache/geode/management/internal/cli/commands/CliCommandTestBase.java +++ b/geode-core/src/test/java/org/apache/geode/management/internal/cli/commands/CliCommandTestBase.java @@ -27,9 +27,9 @@ import java.io.PrintStream; import java.net.InetAddress; import java.net.UnknownHostException; +import java.util.List; import java.util.Map; import java.util.Properties; -import java.util.Set; import java.util.regex.Matcher; import java.util.regex.Pattern; @@ -80,7 +80,7 @@ public static boolean checkIfCommandsAreLoadedOrNot() { CommandManager manager = new CommandManager(); - Set<CommandMarker> commands = manager.getCommandMarkers(); + List<CommandMarker> commands = manager.getCommandMarkers(); return commands.size() >= 1; } diff --git a/geode-core/src/test/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessorTest.java b/geode-core/src/test/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessorTest.java index db7176ba4f..df00cf17aa 100644 --- a/geode-core/src/test/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessorTest.java +++ b/geode-core/src/test/java/org/apache/geode/management/internal/cli/remote/OnlineCommandProcessorTest.java @@ -21,6 +21,8 @@ import static org.mockito.Mockito.mock; import static org.mockito.Mockito.when; +import java.util.Properties; + import org.junit.Before; import org.junit.Test; import org.junit.experimental.categories.Category; @@ -34,6 +36,7 @@ @Category(UnitTest.class) public class OnlineCommandProcessorTest { + Properties properties; SecurityService securityService; CommandExecutor executor; OnlineCommandProcessor onlineCommandProcessor; @@ -41,12 +44,13 @@ @Before public void before() { + properties = new Properties(); securityService = mock(SecurityService.class); executor = mock(CommandExecutor.class); result = mock(Result.class); when(executor.execute(any())).thenReturn(result); - onlineCommandProcessor = new OnlineCommandProcessor(securityService, executor); + onlineCommandProcessor = new OnlineCommandProcessor(properties, securityService, executor); } diff --git a/geode-core/src/test/resources/META-INF/services/org.springframework.shell.core.CommandMarker b/geode-core/src/test/resources/META-INF/services/org.springframework.shell.core.CommandMarker new file mode 100644 index 0000000000..1bb7c6d141 --- /dev/null +++ b/geode-core/src/test/resources/META-INF/services/org.springframework.shell.core.CommandMarker @@ -0,0 +1,8 @@ +# Mock command for org.apache.geode.management.internal.cli.CommandManagerJUnitTest.testLoadPluginCommands +org.apache.geode.management.internal.cli.CommandManagerJUnitTest$MockPluginCommand +# Should cause ServiceConfigurationException while iterating. +#TODO jbarrett - causes tests to be marked as failed because of exception in log +org.apache.geode.management.internal.cli.CommandManagerJUnitTest$MockPluginCommandNotFound + + +# Mock Extension commands +org.apache.geode.internal.cache.extension.mock.MockExtensionCommands \ No newline at end of file diff --git a/geode-lucene/src/main/resources/META-INF/services/org.springframework.shell.core.CommandMarker b/geode-lucene/src/main/resources/META-INF/services/org.springframework.shell.core.CommandMarker new file mode 100644 index 0000000000..0879e15732 --- /dev/null +++ b/geode-lucene/src/main/resources/META-INF/services/org.springframework.shell.core.CommandMarker @@ -0,0 +1,2 @@ +# Lucene Extensions

command +org.apache.geode.cache.lucene.internal.cli.LuceneIndexCommands \ No newline at end of file ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

30. Commit 2b21e2bd6a3ba7fbcd9a2d8e93ae2aef852689ad in geode's branch refs/heads/develop from [~jens.deppe] [https://gitbox.apache.org/repos/asf?p=geode.git;h=2b21e2b] Revert "GEODE-3007: Simplify support for custom GFSH commands (#1042)" (#1072) This reverts commit 0e5dd6ba45519463147337c8265db15e8f1840fc.