Item 184
git_comments:

git_commits:

1. **summary:** GROOVY-9153: fix merge glitch
   message: GROOVY-9153: fix merge glitch

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** StaticInvocationWriter modifies shared global node ConstantExpression.NULL
   description: In the case of a static groovy method call like {{File.createTempDir()}} when {{@CompileStatic}} is applied, {{StaticInvocationWriter}}'s {{writeDirectMethodCall}} and {{loadArguments}} mutate the shared global constant node {{ConstantExpression.NULL}}. Relevant bits: {code:java} protected boolean writeDirectMethodCall(final MethodNode target, final boolean implicitThis, final Expression receiver, final TupleExpression args) { ... if (target instanceof ExtensionMethodNode) { ... if (emn.isStaticExtension()) { // it's a static extension method argumentList.add(0, ConstantExpression.NULL); } else { .... } Parameter[] parameters = node.getParameters(); loadArguments(argumentList, parameters); ... protected void loadArguments(List<Expression> argumentList, Parameter[] para) { ... // This is repeated 3 times in the method: Expression expression = argumentList.get(i); expression.putNodeMetaData(PARAMETER_TYPE, para[i].getType()); {code}
2. **summary:** StaticInvocationWriter modifies shared global node ConstantExpression.NULL
   description: In the case of a static groovy method call like {{File.createTempDir()}} when {{@CompileStatic}} is applied, {{StaticInvocationWriter}}'s {{writeDirectMethodCall}} and {{loadArguments}} mutate the shared global constant node {{ConstantExpression.NULL}}. Relevant bits: {code:java} protected boolean writeDirectMethodCall(final MethodNode target, final boolean implicitThis, final Expression receiver, final TupleExpression args) { ... if (target instanceof ExtensionMethodNode) { ... if (emn.isStaticExtension()) { // it's a static extension method argumentList.add(0, ConstantExpression.NULL); } else { .... } Parameter[] parameters = node.getParameters(); loadArguments(argumentList, parameters); ... protected void loadArguments(List<Expression> argumentList, Parameter[] para) { ... // This is repeated 3 times in the method: Expression expression = argumentList.get(i); expression.putNodeMetaData(PARAMETER_TYPE, para[i].getType()); {code}
3. **summary:** StaticInvocationWriter modifies shared global node ConstantExpression.NULL
   description: In the case of a static groovy method call like {{File.createTempDir()}} when {{@CompileStatic}} is applied, {{StaticInvocationWriter}}'s {{writeDirectMethodCall}} and {{loadArguments}} mutate the shared global constant node {{ConstantExpression.NULL}}. Relevant bits: {code:java} protected boolean writeDirectMethodCall(final MethodNode target, final boolean implicitThis, final Expression receiver, final TupleExpression args) { ... if (target instanceof ExtensionMethodNode) { ... if (emn.isStaticExtension()) { // it's a static extension method argumentList.add(0, ConstantExpression.NULL); } else { .... } Parameter[] parameters = node.getParameters(); loadArguments(argumentList, parameters); ... protected void loadArguments(List<Expression> argumentList, Parameter[] para) { ... // This is repeated 3 times in the method: Expression expression = argumentList.get(i); expression.putNodeMetaData(PARAMETER_TYPE, para[i].getType()); {code}
4. **summary:** StaticInvocationWriter modifies shared global node ConstantExpression.NULL
   description: In the case of a static groovy method call like {{File.createTempDir()}} when {{@CompileStatic}} is applied, {{StaticInvocationWriter}}'s {{writeDirectMethodCall}} and

{{loadArguments}} mutate the shared global constant node {{ConstantExpression.NULL}}. Relevant bits: {code:java} protected boolean writeDirectMethodCall(final MethodNode target, final boolean implicitThis, final Expression receiver, final TupleExpression args) { ... if (target instanceof ExtensionMethodNode) { ... if (emn.isStaticExtension()) { // it's a static extension method argumentList.add(0, ConstantExpression.NULL); } else { .... } Parameter[] parameters = node.getParameters(); loadArguments(argumentList, parameters); ... protected void loadArguments(List<Expression> argumentList, Parameter[] para) { ... // This is repeated 3 times in the method: Expression expression = argumentList.get(i); expression.putNodeMetaData(PARAMETER_TYPE, para[i].getType()); {code}

5. **summary:** StaticInvocationWriter modifies shared global node ConstantExpression.NULL
   description: In the case of a static groovy method call like {{File.createTempDir()}} when {{@CompileStatic}} is applied, {{StaticInvocationWriter}}'s {{writeDirectMethodCall}} and {{loadArguments}} mutate the shared global constant node {{ConstantExpression.NULL}}. Relevant bits: {code:java} protected boolean writeDirectMethodCall(final MethodNode target, final boolean implicitThis, final Expression receiver, final TupleExpression args) { ... if (target instanceof ExtensionMethodNode) { ... if (emn.isStaticExtension()) { // it's a static extension method argumentList.add(0, ConstantExpression.NULL); } else { .... } Parameter[] parameters = node.getParameters(); loadArguments(argumentList, parameters); ... protected void loadArguments(List<Expression> argumentList, Parameter[] para) { ... // This is repeated 3 times in the method: Expression expression = argumentList.get(i); expression.putNodeMetaData(PARAMETER_TYPE, para[i].getType()); {code}
   label: code-design
6. **summary:** StaticInvocationWriter modifies shared global node ConstantExpression.NULL
   description: In the case of a static groovy method call like {{File.createTempDir()}} when {{@CompileStatic}} is applied, {{StaticInvocationWriter}}'s {{writeDirectMethodCall}} and {{loadArguments}} mutate the shared global constant node {{ConstantExpression.NULL}}. Relevant bits: {code:java} protected boolean writeDirectMethodCall(final MethodNode target, final boolean implicitThis, final Expression receiver, final TupleExpression args) { ... if (target instanceof ExtensionMethodNode) { ... if (emn.isStaticExtension()) { // it's a static extension method argumentList.add(0, ConstantExpression.NULL); } else { .... } Parameter[] parameters = node.getParameters(); loadArguments(argumentList, parameters); ... protected void loadArguments(List<Expression> argumentList, Parameter[] para) { ... // This is repeated 3 times in the method: Expression expression = argumentList.get(i); expression.putNodeMetaData(PARAMETER_TYPE, para[i].getType()); {code}
7. **summary:** StaticInvocationWriter modifies shared global node ConstantExpression.NULL
   description: In the case of a static groovy method call like {{File.createTempDir()}} when {{@CompileStatic}} is applied, {{StaticInvocationWriter}}'s {{writeDirectMethodCall}} and {{loadArguments}} mutate the shared global constant node {{ConstantExpression.NULL}}. Relevant bits: {code:java} protected boolean writeDirectMethodCall(final MethodNode target, final boolean implicitThis, final Expression receiver, final TupleExpression args) { ... if (target instanceof ExtensionMethodNode) { ... if (emn.isStaticExtension()) { // it's a static extension method argumentList.add(0, ConstantExpression.NULL); } else { .... } Parameter[] parameters = node.getParameters(); loadArguments(argumentList, parameters); ... protected void loadArguments(List<Expression> argumentList, Parameter[] para) { ... // This is repeated 3 times in the method: Expression expression = argumentList.get(i); expression.putNodeMetaData(PARAMETER_TYPE, para[i].getType()); {code}

jira_issues_comments:

1. To protect the global constant expressions, you could do something like the attached. It's missing an override of addAnnotation.
2. The proposed PR was merged into master and will be merged into 2_5_X later.
3. merged into 2_5_X
4. It is still quite possible for {{StaticInvocationWriter}} to modify one of the shared {{ConstantExpression}} definitions. One example is AutoClone, which creates invokeMethod calls with ConstantExpression.NULL arguments. The test CanonicalComponentsTransformTest hits this case.
5. **body:** This commit should fix off most remaining cases: 73314d6291ff64a00e91fd3cf351cd7ef69cf7cd Some of these aren't strictly needed right now but there is no harm in handling the extra cases and we are now somewhat more evolution proof.

label: code-design