Item 342
git_comments:

1. * Copyright (C) 2015 Google Inc. * * Licensed under the Apache License, Version 2.0 (the "License"); you may not * use this file except in compliance with the License. You may obtain a copy of * the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, WITHOUT * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the * License for the specific language governing permissions and limitations under * the License.
2. Register the (now validated) context info
3. Now, iterate over all the discovered interfaces
4. * * Utility implementing the necessary reflection for working with {@link DoFnWithContext}s.
5. First, find all declared methods on the startClass and parents (up to stopClass)
6. * * Invoke the reflected {@link StartBundle} method on the given instance. * * @param fn an instance of the {@link DoFnWithContext} to invoke {@link StartBundle} on. * @param c the {@link com.google.cloud.dataflow.sdk.transforms.DoFnWithContext.Context} * to pass to {@link StartBundle}.
7. If we have at least one match, then either it should be the only match or it should be an extension of the other matches (which came from parent classes).
8. The DoFnWithContext doesn't allow us to ask for these outside ProcessElements, so this should be unreachable.
9. * * @return true if the reflected {@link DoFnWithContext} uses Keyed State.
10. Exception in user code.
11. Verify that their method arguments satisfy our conditions.
12. * * Invoke the reflected {@link ProcessElement} method on the given instance. * * @param fn an instance of the {@link DoFnWithContext} to invoke {@link ProcessElement} on. * @param c the {@link com.google.cloud.dataflow.sdk.transforms.DoFnWithContext.ProcessContext} * to pass to {@link ProcessElement}.
13. * * Verify the method arguments for a given {@link DoFnWithContext} method. * * <p>The requirements for a method to be valid, are: * * The method has at least one argument. * The first argument is of type firstContextArg. * The remaining arguments have raw types that appear in {@code contexts} * Any generics on the extra context arguments match what is expected. Eg., * {@code WindowingInternals<I, O>} either matches the {@code I} and {@code O} parameters of * the {@code DoFn<I, O>.ProcessContext}, or it uses a wildcard, etc. * * * @param m the method to verify * @param contexts mapping from raw classes to the {@link ExtraContextInfo} used * to create new instances. * @param firstContextArg the expected type of the first context argument * @param iParam TypeParameter representing the input type * @param oParam TypeParameter representing the output type
14. * * @return true if the reflected {@link DoFnWithContext} uses a Single Window.
15. * * Invoke the reflected {@link FinishBundle} method on the given instance. * * @param fn an instance of the {@link DoFnWithContext} to invoke {@link FinishBundle} on. * @param c the {@link com.google.cloud.dataflow.sdk.transforms.DoFnWithContext.Context} * to pass to {@link FinishBundle}.
16. If we get here, the class matches, but maybe the generics don't:
17. * * Create an instance of the given instance using the instance factory.
18. Exception in our code.
19. * * Implementation of {@link DoFnReflector} for the arbitrary {@link DoFnWithContext}.
20. Locate the annotated methods
21. * * @return the {@link DoFnReflector} for the given {@link DoFnWithContext}.
22. * * Create a {@link DoFn} that the {@link DoFnWithContext}.
23. We actually want the owner, since ProcessContext and Context are owned by DoFnWithContext.
24. All of the remaining parameters must be a super-interface of allExtraContextArgs that is not listed in the EXCLUDED_INTERFACES set.
25. We need to be able to call it. We require it is public.
26. * * Create the type token for the given type, filling in the generics.
27. The first parameter must be present, and must be the specified type
28. And make sure its not static.
29. Fill in the generics in the allExtraContextArgs interface from the types in the Context or ProcessContext DoFn.

30. * Copyright (C) 2015 Google Inc. * * Licensed under the Apache License, Version 2.0 (the "License"); you may not * use this file except in compliance with the License. You may obtain a copy of * the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, WITHOUT * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the * License for the specific language governing permissions and limitations under * the License.
31. * * Returns a {@link TypeToken} capturing what is known statically * about the output type of this {@code DoFnWithContext} instance's * most-derived class. * * <p> In the normal case of a concrete {@code DoFnWithContext} subclass with * no generic type parameters of its own (including anonymous inner * classes), this will be a complete non-generic type, which is good * for choosing a default output {@code Coder<O>} for the output * {@code PCollection<O>}.
32. * * Annotation for the method to use for processing elements. A subclass of * {@link DoFnWithContext} must have a method with this annotation satisfying * the following constraints in order for it to be executable: * * It must have at least one argument. * Its first argument must be a {@link DoFnWithContext.ProcessContext}. * Its remaining arguments must be {@link KeyedState}, {@link BoundedWindow}, or * {@link WindowingInternals WindowingInternals<I, O>}. *
33. * * Adds the given element to the side output {@code PCollection} with the * given tag. * * <p> The caller of {@code ParDo} uses {@link ParDo#withOutputTags} to * specify the tags of side outputs that it consumes. Non-consumed side * outputs, e.g., outputs for monitoring purposes only, don't necessarily * need to be specified. * * <p> The output element will have the same timestamp and be in the same * windows as the input element passed to {@link ProcessElement}). * * <p> If invoked from {@link StartBundle} or {@link FinishBundle}, * this will attempt to use the * {@link com.google.cloud.dataflow.sdk.transforms.windowing.WindowFn} * of the input {@code PCollection} to determine what windows the element * should be in, throwing an exception if the {@code WindowFn} attempts * to access any information about the input element. The output element * will have a timestamp of negative infinity. * * @throws IllegalArgumentException if the number of outputs exceeds * the limit of 1,000 outputs per DoFn * @see ParDo#withOutputTags
34. * * Returns a {@link TypeToken} capturing what is known statically * about the input type of this {@code DoFnWithContext} instance's most-derived * class. * * <p> See {@link #getOutputTypeToken} for more discussion.
35. * * Annotation for the method to use to prepare an instance for processing a batch of elements. * The method annotated with this must satisfy the following constraints: * * It must have at least one argument. * Its first (and only) argument must be a {@link DoFnWithContext.Context}. *
36. //
37. * * Construct the {@link KeyedState} interface for use within a {@link DoFnWithContext} that * needs it. This is called if the {@link ProcessElement} method has a parameter of type * {@link KeyedState}. * * @return {@link KeyedState} interface for interacting with keyed state.
38. * * Interface for runner implementors to provide implementations of extra context information. * * <p>The methods on this interface are called by {@link DoFnReflector} before invoking an * annotated {@link StartBundle}, {@link ProcessElement} or {@link FinishBundle} method that * has indicated it needs the given extra context. * * <p>In the case of {@link ProcessElement} it is called once per invocation of * {@link ProcessElement}.
39. * * Returns the {@code PipelineOptions} specified with the * {@link com.google.cloud.dataflow.sdk.runners.PipelineRunner} * invoking this {@code DoFnWithContext}. The {@code PipelineOptions} will * be the default running via {@link DoFnTester}.
40. * * Construct the {@link WindowingInternals} to use within a {@link DoFnWithContext} that * needs it. This is called if the {@link ProcessElement} method has a parameter of type * {@link WindowingInternals}.
41. * * Adds the given element to the main output {@code PCollection}. * * <p> If invoked from {@link ProcessElement}, the output * element will have the same timestamp and be in the same windows * as the input element passed to {@link @ProcessElement}). * * <p> If invoked from {@link StartBundle} or {@link FinishBundle}, * this will attempt to use the * {@link com.google.cloud.dataflow.sdk.transforms.windowing.WindowFn} * of the input {@code PCollection} to determine what windows the element * should be in, throwing an exception if the {@code WindowFn} attempts * to access any information about the input element. The output element * will have a timestamp of negative infinity.
42. * * Returns an {@link Aggregator} with the aggregation logic specified by the * {@link SerializableFunction} argument. The name provided must be unique * across {@link Aggregator}s created within the DoFn. * * @param name the name of the aggregator * @param combiner the {@link

SerializableFunction} to use in the aggregator * @return an aggregator for the provided name and combiner in the scope of * this DoFn * @throws NullPointerException if the name or combiner is null * @throws IllegalArgumentException if the given name collides with another * aggregator in this scope
43. * * Returns the timestamp of the input element. * * <p> See {@link com.google.cloud.dataflow.sdk.transforms.windowing.Window} * for more information.
44. * * Construct the {@link BoundedWindow} to use within a {@link DoFnWithContext} that * needs it. This is called if the {@link ProcessElement} method has a parameter of type * {@link BoundedWindow}. * * @return {@link BoundedWindow} of the element currently being processed.
45. * * Information accessible when running {@link DoFn#processElement}.
46. * * Returns the allowed timestamp skew duration, which is the maximum * duration that timestamps can be shifted backward in * {@link DoFnWithContext.Context#outputWithTimestamp}. * * The default value is {@code Duration.ZERO}, in which case * timestamps can only be shifted forward to future. For infinite * skew, return {@code Duration.millis(Long.MAX_VALUE)}.
47. * * Returns the input element to be processed.
48. * Information accessible to all methods in this {@code DoFnWithContext}.
49. * * Returns the value of the side input. * * @throws IllegalArgumentException if this is not a side input * @see ParDo#withSideInputs
50. * * Returns an {@link Aggregator} with aggregation logic specified by the * {@link CombineFn} argument. The name provided must be unique across * {@link Aggregator}s created within the DoFn. * * @param name the name of the aggregator * @param combiner the {@link CombineFn} to use in the aggregator * @return an aggregator for the provided name and combiner in the scope of * this DoFn * @throws NullPointerException if the name or combiner is null * @throws IllegalArgumentException if the given name collides with another * aggregator in this scope
51. * * Adds the given element to the specified side output {@code PCollection}, * with the given timestamp. * * <p> If invoked from {@link ProcessElement}), the timestamp * must not be older than the input element's timestamp minus * {@link DoFn#getAllowedTimestampSkew}. The output element will * be in the same windows as the input element. * * <p> If invoked from {@link StartBundle} or {@link FinishBundle}, * this will attempt to use the * {@link com.google.cloud.dataflow.sdk.transforms.windowing.WindowFn} * of the input {@code PCollection} to determine what windows the element * should be in, throwing an exception if the {@code WindowFn} attempts * to access any information about the input element except for the * timestamp. * * @throws IllegalArgumentException if the number of outputs exceeds * the limit of 1,000 outputs per DoFn * @see ParDo#withOutputTags
52. * * Adds the given element to the main output {@code PCollection}, * with the given timestamp. * * <p> If invoked from {@link ProcessElement}), the timestamp * must not be older than the input element's timestamp minus * {@link DoFn#getAllowedTimestampSkew}. The output element will * be in the same windows as the input element. * * <p> If invoked from {@link StartBundle} or {@link FinishBundle}, * this will attempt to use the * {@link com.google.cloud.dataflow.sdk.transforms.windowing.WindowFn} * of the input {@code PCollection} to determine what windows the element * should be in, throwing an exception if the {@code WindowFn} attempts * to access any information about the input element except for the * timestamp.
53. * * The argument to {@link ParDo} providing the code to use to process * elements of the input * {@link com.google.cloud.dataflow.sdk.values.PCollection}. * * <p> See {@link ParDo} for more explanation, examples of use, and * discussion of constraints on {@code DoFnWithContext}s, including their * serializability, lack of access to global shared mutable state, * requirements for failure tolerance, and benefits of optimization. * * <p> {@code DoFnWithContext}s can be tested in a particular * {@code Pipeline} by running that {@code Pipeline} on sample input * and then checking its output. Unit testing of a {@code DoFnWithContext}, * separately from any {@code ParDo} transform or {@code Pipeline}, * can be done via the {@link DoFnTester} harness. * * <p>Implementations must define a method annotated with {@link ProcessElement} * that satisfies the requirements described there. See the {@link ProcessElement} * for details. * * <p> This functionality is experimental and likely to change. * * <p> Example usage: * * <pre> {@code * PCollection<String> lines = ... ; * PCollection<String> words = * lines.apply(ParDo.of(new DoFnWithContext<String, String>() { * @ProcessElement * public void processElement(ProcessContext c, BoundedWindow window) { * * * }})); * } </pre> * * @param <I> the type of the (main) input elements * @param <O> the type of the (main) output elements
54. * Copyright (C) 2015 Google Inc. * * Licensed under the Apache License, Version 2.0 (the "License"); you may not * use this file except in compliance with the License. You may obtain a copy of * the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the * License for the specific language governing permissions and limitations under * the License.
55. * * Tests for {@link DoFnReflector}.
56. * Copyright (C) 2015 Google Inc. * * Licensed under the Apache License, Version 2.0 (the "License"); you may not * use this file except in compliance with the License. You may obtain a copy of * the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, WITHOUT * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the * License for the specific language governing permissions and limitations under * the License.
57. * Tests for {@link DoFnWithContext}.
58. * * @param c context
59. * * Returns a {@code DoFnTester} supporting unit-testing of the given * {@link DoFn}.
60. * * Returns a new {@code ParDo} {@code PTransform} that's like this * transform but which will invoke the given {@link DoFnWithContext} * function, and which has its input and output types bound. Does * not modify this transform. The resulting {@code PTransform} is * sufficiently specified to be applied, but more properties can * still be specified.
61. * * Returns a new multi-output {@code ParDo} {@code PTransform} * that's like this transform but which will invoke the given * {@link DoFnWithContext} function, and which has its input type bound. * Does not modify this transform. The resulting * {@code PTransform} is sufficiently specified to be applied, but * more properties can still be specified.
62. * * Creates a {@code ParDo} {@code PTransform} that will invoke the * given {@link DoFnWithContext} function. * * <p> The resulting {@code PTransform}'s types have been bound, with the * input being a {@code PCollection<I>} and the output a * {@code PCollection<O>}, inferred from the types of the argument * {@code DoFn<I, O>}. It is ready to be applied, or further * properties can be set on it first. * * <p> {@link DoFnWithContext} is an experimental alternative to * {@link DoFn} which simplifies accessing {@code KeyedState} and * the window of the element.
63. * * Returns all interfaces of the given clazz. * @param clazz * @return
64. * * Returns all the methods visible from the provided interfaces. * * @param interfaces The interfaces to use when searching for all their methods. * @return An iterable of {@link Method}s which interfaces expose.
65. * * Returns all the methods visible from {@code iface}. * * @param iface The interface to use when searching for all its methods. * @return An iterable of {@link Method}s which {@code iface} exposes.

git_commits:

1. **summary:** Introduce DoFnWithContext, an annotation based version of DoFn.
   message: Introduce DoFnWithContext, an annotation based version of DoFn. Current implementation is via DoFnReflector and an adaptor to turn a DoFnWithContext into a DoFn. ----Release Notes---- Introduce DoFnWithContext, an experimental way to simplify accessing extra information such as KeyedState and the current window for a DoFn. Consider using DoFnWithContext rather than creating a DoFn that implements the RequiresKeyedState or RequiresWindowAccess. [] ------------- Created by MOE: http://code.google.com/p/moe-java MOE_MIGRATED_REVID=92189663

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

jira_issues_comments: