

# Sample 10 modified code.

[Browse files](#)


🔗 master

 **mjdecker** committed 8 hours ago

1 parent [4378bad](#) commit [6e80e8be3271e1a827aaba305d8015c0802aeb28](#)

 Showing **1 changed file** with **44 additions** and **36 deletions**.

[Unified](#) [Split](#)

80  tool1/sample10.java [View](#)

...	@@ -1,25 +1,27 @@	
1	package com.google.common.cache;	1 package com.google.common.cache;
2		2
3	import com.google.common.base.Function;	3 import com.google.common.base.Function;
4	import com.google.common.base.Objects;	4 import com.google.common.base.Objects;
5	import com.google.common.base.Optional;	5 import com.google.common.base.Optional;
6	import com.google.common.base.Preconditions;	6 import com.google.common.base.Preconditions;
7	import com.google.common.cache.LocalCache.Strength;	7 import com.google.common.cache.LocalCache.Strength;
8	import com.google.common.collect.Iterables;	8 import com.google.common.collect.Iterables;
9	import com.google.common.collect.Lists;	9 import com.google.common.collect.Lists;
10	import com.google.common.collect.Sets;	10 import com.google.common.collect.Sets;
11		11
12	import java.util.List;	12 import java.util.List;
13	import java.util.Set;	13 import java.util.Set;
14	import java.util.concurrent.TimeUnit;	14 import java.util.concurrent.TimeUnit;
15		15
16	import javax.annotation.Nullable;	16 import javax.annotation.Nullable;
17		17
18	class CacheBuilderFactory {	18 class CacheBuilderFactory {
19	private Set<Integer> concurrencyLevels =	19 private Set<Integer> concurrencyLevels =
	Sets.newHashSet((Integer) null);	Sets.newHashSet((Integer) null);
20	private Set<Integer> initialCapacities =	20 private Set<Integer> initialCapacities =
	Sets.newHashSet((Integer) null);	Sets.newHashSet((Integer) null);
21	private Set<Integer> maximumSizes =	21 private Set<Integer> maximumSizes =
	Sets.newHashSet((Integer) null);	Sets.newHashSet((Integer) null);
22	- private Set<ExpirationSpec> expirations =	22 + private Set<DurationSpec> expireAfterWrites =
	Sets.newHashSet((ExpirationSpec) null);	Sets.newHashSet((DurationSpec) null);
		23 + private Set<DurationSpec> expireAfterAccesses =
		Sets.newHashSet((DurationSpec) null);
		24 + private Set<DurationSpec> refreshes =
		Sets.newHashSet((DurationSpec) null);
23	private Set<Strength> keyStrengths =	25 private Set<Strength> keyStrengths =
	Sets.newHashSet((Strength) null);	Sets.newHashSet((Strength) null);
24	private Set<Strength> valueStrengths =	26 private Set<Strength> valueStrengths =
	Sets.newHashSet((Strength) null);	Sets.newHashSet((Strength) null);
25		27
26	CacheBuilderFactory withConcurrencyLevels(Set<Integer>	28 CacheBuilderFactory withConcurrencyLevels(Set<Integer>
	concurrencyLevels) {	concurrencyLevels) {
27	this.concurrencyLevels =	29 this.concurrencyLevels =
	Sets.newLinkedHashSet(concurrencyLevels);	Sets.newLinkedHashSet(concurrencyLevels);
28	return this;	30 return this;
29	}	31 }
30		32
31	CacheBuilderFactory withInitialCapacities(Set<Integer>	33 CacheBuilderFactory withInitialCapacities(Set<Integer>
	initialCapacities) {	initialCapacities) {
32	this.initialCapacities =	34 this.initialCapacities =
	Sets.newLinkedHashSet(initialCapacities);	Sets.newLinkedHashSet(initialCapacities);
33	return this;	35 return this;
34	}	36 }
35		37
36	CacheBuilderFactory withMaximumSizes(Set<Integer>	38 CacheBuilderFactory withMaximumSizes(Set<Integer>

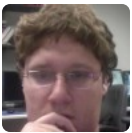
37	<code>maximumSizes) {     this.maximumSizes = Sets.newLinkedHashSet(maximumSizes);</code>	39	<code>maximumSizes) {     this.maximumSizes = Sets.newLinkedHashSet(maximumSizes);</code>
38	<code>    return this;</code>	40	<code>    return this;</code>
39	<code>}</code>	41	<code>}</code>
40		42	
41	<code>- CacheBuilderFactory withExpirations(Set&lt;ExpirationSpec&gt; expirations) {</code>	43	<code>+ CacheBuilderFactory withExpireAfterWrites(Set&lt;DurationSpec&gt; durations) {</code>
42	<code>-    this.expirations = Sets.newLinkedHashSet(expirations);</code>	44	<code>+    this.expireAfterWrites = Sets.newLinkedHashSet(durations);</code>
		45	<code>+    return this;</code>
		46	<code>+ }</code>
		47	<code>+ </code>
		48	<code>+ CacheBuilderFactory withExpireAfterAccesses(Set&lt;DurationSpec&gt; durations) {</code>
		49	<code>+    this.expireAfterAccesses = Sets.newLinkedHashSet(durations);</code>
		50	<code>+    return this;</code>
		51	<code>+ }</code>
		52	<code>+ </code>
		53	<code>+ CacheBuilderFactory withRefreshes(Set&lt;DurationSpec&gt; durations) {</code>
		54	<code>+    this.refreshes = Sets.newLinkedHashSet(durations);</code>
43	<code>    return this;</code>	55	<code>    return this;</code>
44	<code>}</code>	56	<code>}</code>
45		57	
46	<code>CacheBuilderFactory withKeyStrengths(Set&lt;Strength&gt; keyStrengths) {</code>	58	<code>CacheBuilderFactory withKeyStrengths(Set&lt;Strength&gt; keyStrengths) {</code>
47	<code>    this.keyStrengths = Sets.newLinkedHashSet(keyStrengths);</code>	59	<code>    this.keyStrengths = Sets.newLinkedHashSet(keyStrengths);</code>
48	<code>Preconditions.checkArgument(!this.keyStrengths.contains(Streng th.SOFT));</code>	60	<code>Preconditions.checkArgument(!this.keyStrengths.contains(Streng th.SOFT));</code>
49	<code>    return this;</code>	61	<code>    return this;</code>
50	<code>}</code>	62	<code>}</code>
51		63	
52	<code>CacheBuilderFactory withValueStrengths(Set&lt;Strength&gt; valueStrengths) {</code>	64	<code>CacheBuilderFactory withValueStrengths(Set&lt;Strength&gt; valueStrengths) {</code>
53	<code>    this.valueStrengths = Sets.newLinkedHashSet(valueStrengths);</code>	65	<code>    this.valueStrengths = Sets.newLinkedHashSet(valueStrengths);</code>
54	<code>    return this;</code>	66	<code>    return this;</code>
55	<code>}</code>	67	<code>}</code>
56		68	
57	<code>Iterable&lt;CacheBuilder&lt;Object, Object&gt;&gt; buildAllPermutations() {</code>	69	<code>Iterable&lt;CacheBuilder&lt;Object, Object&gt;&gt; buildAllPermutations() {</code>
58	<code>    @SuppressWarnings("unchecked")</code>	70	<code>    @SuppressWarnings("unchecked")</code>
59	<code>    Iterable&lt;List&lt;Object&gt;&gt; combinations = buildCartesianProduct(concurrencyLevels,</code>	71	<code>    Iterable&lt;List&lt;Object&gt;&gt; combinations = buildCartesianProduct(concurrencyLevels,</code>
60	<code>-    initialCapacities, maximumSizes, expirations, keyStrengths, valueStrengths);</code>	72	<code>+    initialCapacities, maximumSizes, expireAfterWrites, expireAfterAccesses, refreshes,</code>
		73	<code>+    keyStrengths, valueStrengths);</code>
61	<code>    return Iterables.transform(combinations,</code>	74	<code>    return Iterables.transform(combinations,</code>
62	<code>        new Function&lt;List&lt;Object&gt;, CacheBuilder&lt;Object, Object&gt;&gt;&gt;() {</code>	75	<code>        new Function&lt;List&lt;Object&gt;, CacheBuilder&lt;Object, Object&gt;&gt;&gt;() {</code>
63	<code>            @Override public CacheBuilder&lt;Object, Object&gt; apply(List&lt;Object&gt; combination) {</code>	76	<code>            @Override public CacheBuilder&lt;Object, Object&gt; apply(List&lt;Object&gt; combination) {</code>
64	<code>                return createCacheBuilder(                     (Integer) combination.get(0),                     (Integer) combination.get(1),                     (Integer) combination.get(2),                     (ExpirationSpec) combination.get(3),                     (Strength) combination.get(4),                     (Strength) combination.get(5));</code>	77	<code>                return createCacheBuilder(                     (Integer) combination.get(0),                     (Integer) combination.get(1),                     (Integer) combination.get(2),                     (DurationSpec) combination.get(3),                     (DurationSpec) combination.get(4),                     (DurationSpec) combination.get(5),                     (Strength) combination.get(6),                     (Strength) combination.get(7));</code>
65		78	
66		79	
67		80	
68		81	
69		82	
70		83	
		84	
		85	
71	<code>        }</code>	86	<code>        }</code>
72	<code>    });</code>	87	<code>    });</code>
73	<code>}</code>	88	<code>}</code>
74		89	
75	<code>private static final Function&lt;Object, Optional&lt;?&gt;&gt; NULLABLE_TO_OPTIONAL =     new Function&lt;Object, Optional&lt;?&gt;&gt;() {</code>	90	<code>private static final Function&lt;Object, Optional&lt;?&gt;&gt; NULLABLE_TO_OPTIONAL =     new Function&lt;Object, Optional&lt;?&gt;&gt;() {</code>
76	<code>        @Override public Optional&lt;?&gt; apply(@Nullable Object obj) {</code>	91	<code>        @Override public Optional&lt;?&gt; apply(@Nullable Object obj) {</code>
77	<code>            return Optional.fromNullable(obj);</code>	92	<code>            return Optional.fromNullable(obj);</code>
78		93	

79	}	94	}
80	};	95	};
81		96	
82	private static final Function<Optional<?>, Object>	97	private static final Function<Optional<?>, Object>
83	OPTIONAL_TO_NULLABLE =	98	OPTIONAL_TO_NULLABLE =
84	new Function<Optional<?>, Object>() {	99	new Function<Optional<?>, Object>() {
85	@Override public Object apply(Optional<?> optional) {	100	@Override public Object apply(Optional<?> optional) {
86	return optional.orNull();	101	return optional.orNull();
87	}	102	}
88	};	103	};
89	private Iterable<List<Object>> buildCartesianProduct(Set<?	104	private Iterable<List<Object>> buildCartesianProduct(Set<?
90	>... sets) {	105	>... sets) {
91	List<Set<Optional<?>>> optionalSets =	106	List<Set<Optional<?>>> optionalSets =
92	Lists.newArrayListWithExpectedSize(sets.length);	107	Lists.newArrayListWithExpectedSize(sets.length);
93	for (Set<?> set : sets) {	108	for (Set<?> set : sets) {
94	Set<Optional<?>> optionalSet =	109	Set<Optional<?>> optionalSet =
95	Sets.newLinkedHashSet(Iterables.transform(set,	110	Sets.newLinkedHashSet(Iterables.transform(set,
96	NULLABLE_TO_OPTIONAL));	111	NULLABLE_TO_OPTIONAL));
97	optionalSets.add(optionalSet);	112	optionalSets.add(optionalSet);
98	}	113	}
99	Set<List<Optional<?>>> cartesianProduct =	114	Set<List<Optional<?>>> cartesianProduct =
100	Sets.cartesianProduct(optionalSets);	115	Sets.cartesianProduct(optionalSets);
101	return Iterables.transform(cartesianProduct,	116	return Iterables.transform(cartesianProduct,
102	new Function<List<Optional<?>>, List<Object>>() {	117	new Function<List<Optional<?>>, List<Object>>() {
103	@Override public List<Object> apply(List<Optional<?	118	@Override public List<Object> apply(List<Optional<?
104	>> objs) {	119	>> objs) {
105	return Lists.transform(objs,	120	return Lists.transform(objs,
106	OPTIONAL_TO_NULLABLE);	121	OPTIONAL_TO_NULLABLE);
107	}	122	}
108	});	123	});
109	});	124	});
110	}	125	}
111		126	
112		127	
113		128	
114		129	
115		130	
116		131	
117		132	
118		133	
119		134	
120		135	
121		136	
122			
123		137	
124		138	
125		139	
126		140	
127		141	
128		142	
129		143	
130		144	
131		145	
132		146	
		147	
		148	
		149	

133	<code>return builder;</code>	150	<code>return builder;</code>
134	<code>}</code>	151	<code>}</code>
135		152	
136	<code>- static class ExpirationSpec {</code>	153	<code>+ static class DurationSpec {</code>
137	<code>- @Nullable</code>	154	<code>+ private final long duration;</code>
138	<code>- private final Long expireAfterAccessMillis;</code>	155	<code>+ private final TimeUnit unit;</code>
139	<code>- @Nullable</code>		
140	<code>- private final Long expireAfterWriteMillis;</code>		
141	<code>-</code>		
142	<code>- private ExpirationSpec(Long expireAfterAccessMillis, Long expireAfterWriteMillis) {</code>		
143	<code>- Preconditions.checkArgument(</code>		
144	<code>- expireAfterAccessMillis == null   </code>		
145	<code>expireAfterWriteMillis == null);</code>		
146	<code>- this.expireAfterAccessMillis = expireAfterAccessMillis;</code>		
147	<code>- this.expireAfterWriteMillis = expireAfterWriteMillis;</code>		
148	<code>- }</code>		
149	<code>- public static ExpirationSpec afterAccess(long afterAccess, TimeUnit unit) {</code>	156	
150	<code>- return new ExpirationSpec(unit.toMillis(afterAccess), null);</code>	157	<code>+ private DurationSpec(long duration, TimeUnit unit) {</code>
		158	<code>+ this.duration = duration;</code>
		159	<code>+ this.unit = unit;</code>
151	<code>}</code>	160	<code>}</code>
152		161	
153	<code>- public static ExpirationSpec afterWrite(long afterWrite, TimeUnit unit) {</code>	162	<code>+ public static DurationSpec of(long duration, TimeUnit unit) {</code>
154	<code>- return new ExpirationSpec(null, unit.toMillis(afterWrite));</code>	163	<code>+ return new DurationSpec(duration, unit);</code>
155	<code>}</code>	164	<code>}</code>
156		165	
157	<code>@Override</code>	166	<code>@Override</code>
158	<code>public int hashCode() {</code>	167	<code>public int hashCode() {</code>
159	<code>- return Objects.hashCode(expireAfterAccessMillis, expireAfterWriteMillis);</code>	168	<code>+ return Objects.hashCode(duration, unit);</code>
160	<code>}</code>	169	<code>}</code>
161		170	
162	<code>@Override</code>	171	<code>@Override</code>
163	<code>public boolean equals(Object o) {</code>	172	<code>public boolean equals(Object o) {</code>
164	<code>- if (o instanceof ExpirationSpec) {</code>	173	<code>+ if (o instanceof DurationSpec) {</code>
165	<code>- ExpirationSpec that = (ExpirationSpec) o;</code>	174	<code>+ DurationSpec that = (DurationSpec) o;</code>
166	<code>- return Objects.equal(this.expireAfterAccessMillis, that.expireAfterAccessMillis)</code>	175	<code>+ return unit.toNanos(duration) ==</code>
167	<code>- &amp;&amp; Objects.equal(this.expireAfterWriteMillis, that.expireAfterWriteMillis);</code>		<code>that.unit.toNanos(that.duration);</code>
168	<code>}</code>	176	<code>}</code>
169	<code>return false;</code>	177	<code>return false;</code>
170	<code>}</code>	178	<code>}</code>
171		179	
172	<code>@Override</code>	180	<code>@Override</code>
173	<code>public String toString() {</code>	181	<code>public String toString() {</code>
174	<code>return Objects.toStringHelper(this)</code>	182	<code>return Objects.toStringHelper(this)</code>
175	<code>- .add("expireAfterAccessMillis", expireAfterAccessMillis)</code>	183	<code>+ .add("duration", duration)</code>
176	<code>- .add("expireAfterWriteMillis", expireAfterWriteMillis)</code>	184	<code>+ .add("unit", unit)</code>
177	<code>.toString();</code>	185	<code>.toString();</code>
178	<code>}</code>	186	<code>}</code>
179	<code>}</code>	187	<code>}</code>
180	<code>}</code>	188	<code>}</code>

0 comments on commit 6e80e8b

🔒 Lock conversation



Write

Preview

AA ▾ B i “ <> 🔗 ⋮ ½ ⅓ ✓ @ ★



Leave a comment

Attach files by dragging & dropping or [selecting them](#).

 Styling with Markdown is supported

Comment on this commit



 **Unsubscribe**

You're receiving notifications because you're subscribed to this repository.

