

RANDOMIZED TESTING

Solr & Lucene use case

Dawid WEISS

Carrot Search, Poland

Geecon Conference
Poznań, May 2012



Dawid Weiss

Likes coding

10 years assembly only

Likes research

Former academic. PhD in IR.

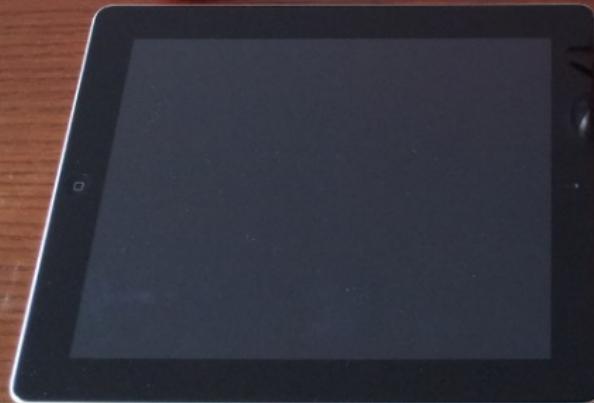
Likes open source

Carrot², HPPC, Lucene, ...

Likes industry

Carrot Search s.c.

How to make this presentation interesting?





APad



Talk outline

Test randomization is cool.

...and it's good for you.

Randomized testing.

...what can be randomized, how to assert on the unknown.

Support from tools.

RandomizedTesting package.

Parallel <junit>.

How to speed up tests on multicores.

Unit Testing

The things we all have been told about

Unit Testing

Tests are good.

More tests → more reliable software.

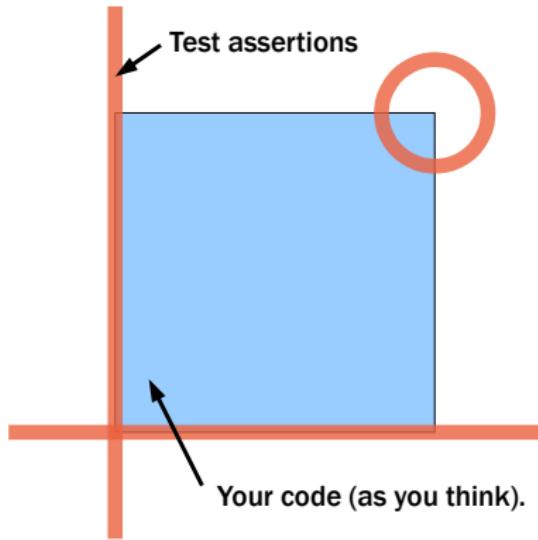
Tests should cover boundary conditions.

But...

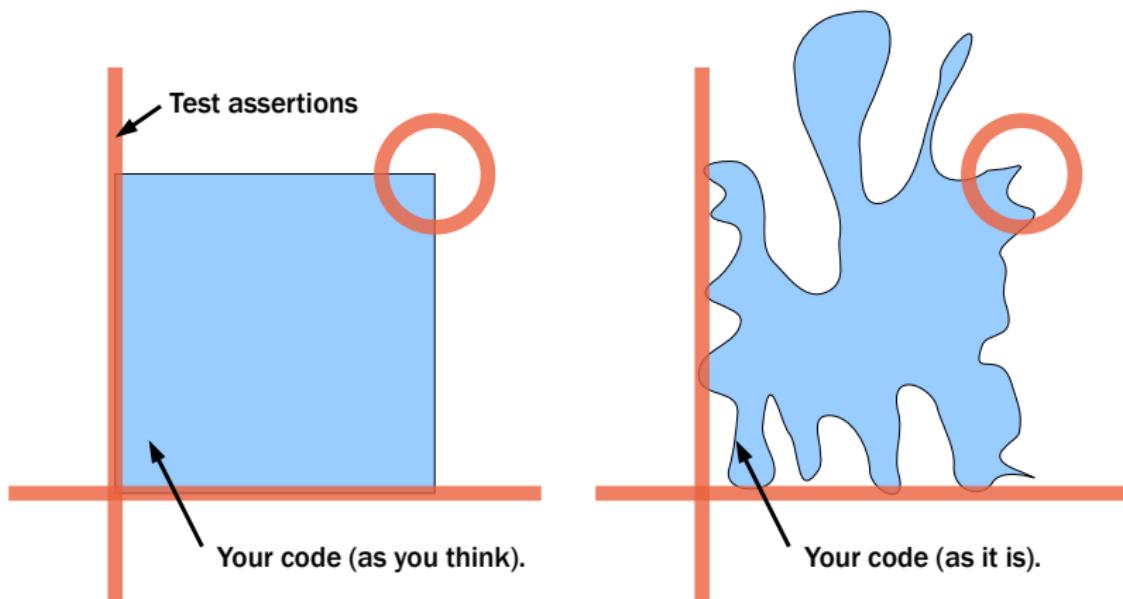
1 execution = 1000 executions.

Tests react to changes (regressions) only.

isRectangle(shape)



isRectangle(shape)



But...

1 execution = 1000 executions.

Tests react to changes (regressions) only.

Complex interactions are hard implement.

Boundary conditions in complex code are rarely or never reached.

But...

1 execution = 1000 executions.

Tests react to changes (regressions) only.

Complex interactions are hard implement.

Boundary conditions in complex code are rarely or never reached.

Increased CO₂ emission :)

Wasted CPU cycles on CI/build servers, developer test runs.

Example.

Can this ever fail?

```
String [] words = {  
    "Poznan", "Spring", "sun" };  
public void poznanWords(Random r) {  
    for (int i = 0; i < 100; i++) {  
        int index =  
            Math.abs(r.nextInt()) % words.length;  
        System.out.println(words[index]);  
    }  
}
```

Example. Can this ever fail?

```
String [] words = {  
    "Poznan", "Spring", "sun" };  
public void poznanWords(Random r) {  
    for (int i = 0; i < 100; i++) {  
        int index =  
            Math.abs(r.nextInt()) % words.length;  
        System.out.println(words[index]);  
    }  
}
```

Happy coders
keep it green...
or something.

```
@Test  
public void testPoznanWords() {  
    poznanWords(new Random());  
}
```

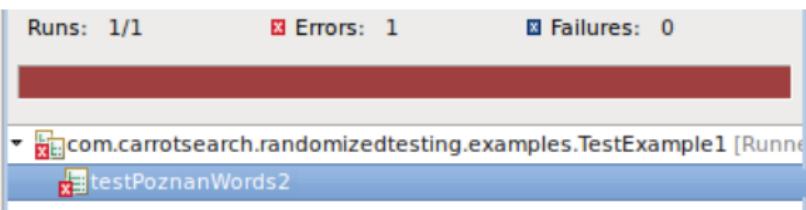
Example. Can this ever fail?

```
String [] words = {"Poznan", "Spring", "sun" };
public void poznanWords(Random r) {
    for (int i = 0; i < 100; i++) {
        int index =
            Math.abs(r.nextInt()) % words.length;
        System.out.println(words[index]);
    }
}
```

Happy coders
keep it green...
or something.

But once in 2^{32} ...

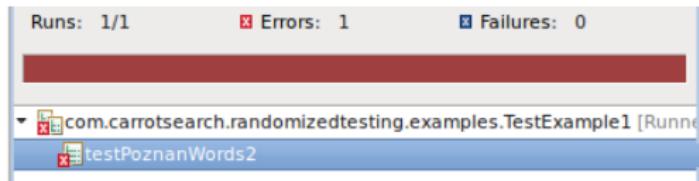
```
@Test
public void testPoznanWords() {
    poznanWords(new Random());
}
```



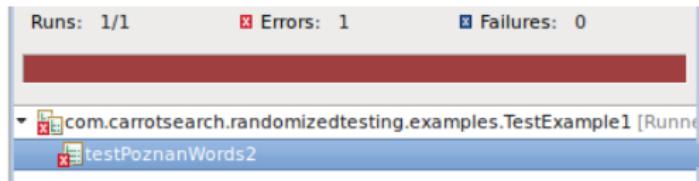
The why?

How do you debug it?

How do you repeat it?



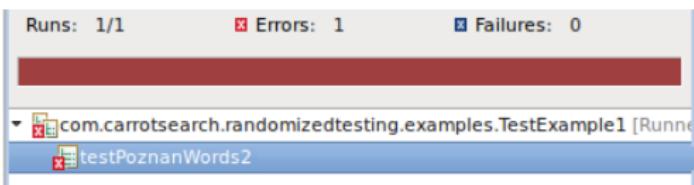
**How do you debug it?
How do you repeat it?**



**Use pseudo-randomness
and publish the seed.**

**How do you debug it?
How do you repeat it?**

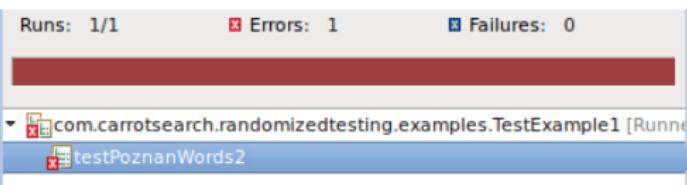
**Use pseudo-randomness
and publish the seed.**



```
@Test
@Seed("487A51B") // get unlucky...
public void testPoznanWords2() {
    poznanWords(
        RandomizedContext.current()
            .getRandom());
}
```

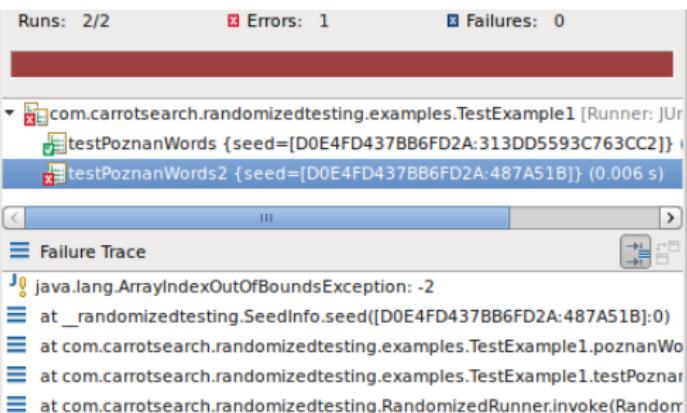
**How do you debug it?
How do you repeat it?**

**Use pseudo-randomness
and publish the seed.**



```
@Test
@Seed("487A51B") // get unlucky...
public void testPoznanWords2() {
    poznanWords(
        RandomizedContext.current()
            .getRandom());
}
```

Crashes every time.



Boundary conditions are everywhere.

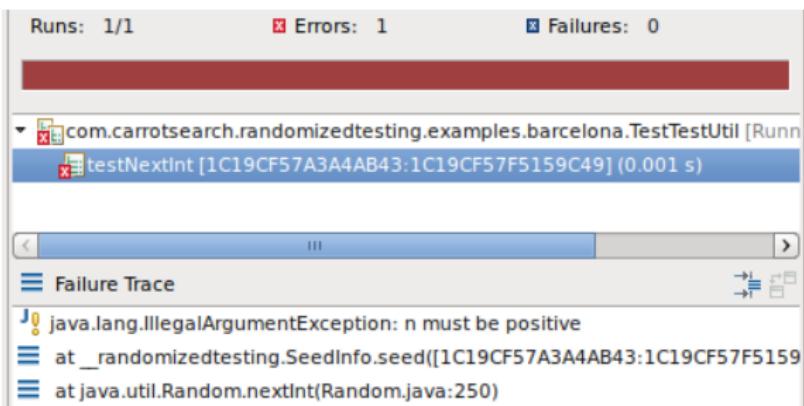
```
/** From: Lucene's _TestUtil.  
 * Start and end are BOTH inclusive. */  
int nextInt(Random r, int start, int end) {  
    return start + r.nextInt(end - start + 1);  
}
```

Boundary conditions are everywhere.

```
/** From: Lucene's _TestUtil.  
 * Start and end are BOTH inclusive. */  
int nextInt(Random r, int start, int end) {  
    return start + r.nextInt(end - start + 1);  
}
```

Gotcha!

```
nextInt(r, 0, Integer.MAX_VALUE)
```



Randomized Testing

Randomized Testing

(mini-manifesto)

Randomized Testing

(mini-manifesto)

Each test covers a possibly different execution path (or data).

Randomized Testing

(mini-manifesto)

Each test covers a possibly different execution path (or data).
Each test can be repeated given the same randomization seed.

Randomized Testing

(mini-manifesto)

- Each test covers a possibly different execution path (or data).
- Each test can be repeated given the same randomization seed.
- Each test is repeated lots of times (→ build server, developers).

Historical note

Industry and academia

Duran, Ntafos: An evaluation of random testing, 1984! Haskell: QuickCheck.

Hacking.

Fuzzifiers, vulnerability discovery.

Lucene Test Framework.

Lots of great ideas. Driven by real needs and bugs. Many contributors.

Carrot Search.

Handcrafted pseudo-randomness in Carrot2, Lingo3G, HPPC, ...

What can be randomized?

1. Input data, iteration counts, arguments.

Random, constraint-bound, shuffled.

```
// Lucene/Solr source code (LTC)
final int iters = atLeast(200);
for (int iter = 0; iter < iters; iter++) {
    // ...
}
```

```
// Lucene/Solr source code (LTC)
if (random.nextBoolean()) {
    term = _TestUtil.randomRealisticUnicodeString(random);
} else {
    // we want to mix in limited-alphabet symbols so
    // we get more sharing of the nodes given how few
    // terms we are testing...
    term = simpleRandomString(random);
}
```

What can be randomized?

1. Input data, iteration counts, arguments.

Random, constraint-bound, shuffled.

2. Software components.

If multiple implementations exist. LTC: Field, Directory, IndexSearcher...

```
public static String randomDirectory(Random random) {  
    if (rarely(random)) {  
        return CORE_DIRECTORIES[random.nextInt(CORE_DIRECTORIES.length)];  
    } else {  
        return "RAMDirectory";  
    }  
}
```

What can be randomized?

1. Input data, iteration counts, arguments.

Random, constraint-bound, shuffled.

2. Software components.

If multiple implementations exist. LTC: Field, Directory, IndexSearcher...

3. Environment.

Locale, TimeZone, JVM (!), operating system.

```
locale = TEST_LOCALE.equals("random") ? randomLocale(random) : ...
Locale.setDefault(locale);
timeZone = TEST_TIMEZONE.equals("random") ? randomTimeZone(random) : ...
TimeZone.setDefault(timeZone);
```

What can be randomized?

1. Input data, iteration counts, arguments.

Random, constraint-bound, shuffled.

2. Software components.

If multiple implementations exist. LTC: Field, Directory, IndexSearcher...

3. Environment.

Locale, TimeZone, JVM (!), operating system.

4. Exceptional triggers.

I/O problems, network problems (using mocks or runtime engineering).

What can be asserted?

Exact output.

If > 1 method is available; naïve algorithms, different implementations.

What can be asserted?

Exact output.

If > 1 method is available; naïve algorithms, different implementations.

Sanity checks.

Only crude output checks and assertions inside the codebase. Logs.

What can be asserted?

Exact output.

If > 1 method is available; naïve algorithms, different implementations.

Sanity checks.

Only crude output checks and assertions inside the codebase. Logs.

Nothing.

Waiting for an exception. Or a jvm core dump. Surprisingly effective :)

RandomizedRunner

Writing randomized tests

Home-grown solutions.

At least log the seed somehow...

```
Long seed = System.currentTimeMillis();
System.out.println(seed);
Random rnd = new Random(seed);
passToTests(rnd);
```

Writing randomized tests

Home-grown solutions.

At least log the seed somehow...

```
long seed = System.currentTimeMillis();
System.out.println(seed);
Random rnd = new Random(seed);
passToTests(rnd);
```

RandomizedRunner.

LUCENE-3492. Written from scratch
in an attempt to decouple randomized
testing from Lucene's internals.

```
@RunWith(RandomizedRunner.class)
public class MyRandomizedTest
```

RandomizedRunner's goals

RandomizedRunner's goals

Compatibility

with JUnit (and tools). At 99%, relax contracts when useful.

RandomizedRunner's goals

Compatibility

with JUnit (and tools). At 99%, relax contracts when useful.

Built-in randomization

including reporting/ stack augmentations.

RandomizedRunner's goals

Compatibility

with JUnit (and tools). At 99%, relax contracts when useful.

Built-in randomization

including reporting/ stack augmentations.

Test isolation

by tracking spawned threads. Timeouts. Terminations.

RandomizedRunner's goals

Compatibility

with JUnit (and tools). At 99%, relax contracts when useful.

Built-in randomization

including reporting/ stack augmentations.

Test isolation

by tracking spawned threads. Timeouts. Terminations.

Utilities

@Repeat, @Seed, @Nightly, @TestGroup, @TestFactories...

@RR: Basics

Using RandomizedRunner

```
@RunWith(RandomizedRunner.class)
public class TestSomething {
    @Test
    public void myTest() {
        // your code here.
    }
}
```

99% of situations identical to the default runner.

Using RandomizedRunner

```
@RunWith(RandomizedRunner.class)
public class TestSomething {
    @Test
    public void myTest() {
        // your code here.
    }
}
```

99% of situations identical to the default runner.
Failures typically a result of incorrect assumptions!

```
@RunWith(RandomizedRunner.class)
public class TestClass1 {
    @BeforeClass private static void beforeClass() { println("class 1: beforeClass"); }
    @Before    private void before()        { println("  class 1: before"); }
    @Test     public void test1_1()        { println("    class 1: test1"); }
    @Test     public void test1_2()        { println("    class 1: test2"); }
    @Test     public void test1_3()        { println("    class 1: test3"); }
    @After    private void after()        { println("  class 1: after"); }
    @AfterClass private static void afterClass() { println("class 1: afterClass"); }
}

public class TestClass2 extends TestClass1 {
    @BeforeClass private static void beforeClass() { println("class 2: beforeClass"); }
    @Before    private void before()        { println("  class 2: before"); }
    @Test     public void test2_1()        { println("    class 2: test1"); }
    @Test     public void test2_2()        { println("    class 2: test2"); }
    @Test     public void test2_3()        { println("    class 2: test3"); }
    @After    private void after()        { println("  class 2: after"); }
    @AfterClass private static void afterClass() { println("class 2: afterClass"); }
}
```

```
class 1: beforeClass
class 2: beforeClass
  class 1: before
  class 2: before
    class 1: test3
class 2: after
class 1: after
class 1: before
class 2: before
  class 2: test1
class 2: after
class 1: after
class 1: before
class 2: before
  class 2: test2
class 2: after
class 1: after
class 1: before
class 2: before
  class 1: test1
class 2: after
class 1: after
class 1: before
class 2: before
  class 2: test3
class 2: after
class 1: after
class 1: before
class 2: before
  class 1: test2
class 2: after
class 1: after
class 1: after
class 2: afterClass
class 1: afterClass
```

```
class 1: beforeClass
class 2: beforeClass
class 1: before
class 2: before
    class 2: test2
class 2: after
class 1: after
class 1: before
class 2: before
    class 1: test2
class 2: after
class 1: after
class 1: before
class 2: before
    class 2: test3
class 2: after
class 1: after
class 1: before
class 2: before
    class 2: test1
class 2: after
class 1: after
class 1: before
class 2: before
    class 1: test1
class 2: after
class 1: after
class 1: before
class 2: before
    class 1: test3
class 2: after
class 1: after
class 1: after
class 2: afterClass
class 1: afterClass
```

```
class 1: beforeClass
class 2: beforeClass
class 1: before
class 2: before
    class 1: test1
class 2: after
class 1: after
class 1: before
class 2: before
    class 2: test3
class 2: after
class 1: after
class 1: before
class 2: before
    class 2: test2
class 2: after
class 1: after
class 1: before
class 2: before
    class 1: test3
class 2: after
class 1: after
class 1: before
class 2: before
    class 2: test1
class 2: after
class 1: after
class 1: before
class 2: before
    class 1: test2
class 2: after
class 1: after
class 1: before
class 2: afterClass
class 1: afterClass
```

▼ com.carrotsearch.randomizedtesting.examples.TestClass2 [Runner: JUnit 4]

- test2_3 {seed=[68508C8145637F4:6F5AC393F4DC201B]} (0.013 s)
- test1_2 {seed=[68508C8145637F4:21C04FFE26CC71AA]} (0.008 s)
- test2_1 {seed=[68508C8145637F4:4F3143B3C6BCCC21]} (0.004 s)
- test1_1 {seed=[68508C8145637F4:C8D97BD78E4607BF]} (0.003 s)
- test2_2 {seed=[68508C8145637F4:CF0918A9EF5A933F]} (0.003 s)
- test1_3 {seed=[68508C8145637F4:44418D9217A50763]} (0.004 s)

▼ com.carrotsearch.randomizedtesting.examples.TestClass2 [Runner: JUnit 4]

- test1_2 {seed=[98233F305B18EE47:BF6678066982A819]} (0.013 s)
- test1_1 {seed=[98233F305B18EE47:567F4C2FC108DE0C]} (0.007 s)
- test1_3 {seed=[98233F305B18EE47:DAE7BA6A58EBDED0]} (0.005 s)
- test2_3 {seed=[98233F305B18EE47:F1FCF46BBB92F9A8]} (0.004 s)
- test2_2 {seed=[98233F305B18EE47:51AF2F51A0144A8C]} (0.003 s)
- test2_1 {seed=[98233F305B18EE47:D197744B89F21592]} (0.005 s)

▼ com.carrotsearch.randomizedtesting.examples.TestClass2 [Runner: JUnit 4]

- test2_3 {seed=[90E59ACD5471DAC4:F93A5196B4FBCD2B]} (0.012 s)
- test1_3 {seed=[90E59ACD5471DAC4:D2211F975782EA53]} (0.008 s)
- test1_2 {seed=[90E59ACD5471DAC4:B7A0DDFB66EB9C9A]} (0.005 s)
- test2_2 {seed=[90E59ACD5471DAC4:59698AACAF7D7E0F]} (0.004 s)
- test2_1 {seed=[90E59ACD5471DAC4:D951D1B6869B2111]} (0.003 s)
- test1_1 {seed=[90E59ACD5471DAC4:5EB9E9D2CE61EA8F]} (0.006 s)

Touching Randomness

Randomized context.

Thread-bound. Defined per lifecycle.
(suite-level, method-level).

```
RandomizedContext ctx =  
    RandomizedContext.current();  
ctx.getTargetClass()  
ctx.getRandom() // => no setSeed()  
ctx.getRandomness().getSeed()  
ctx.isNightly()  
...
```

Touching Randomness

Randomized context.

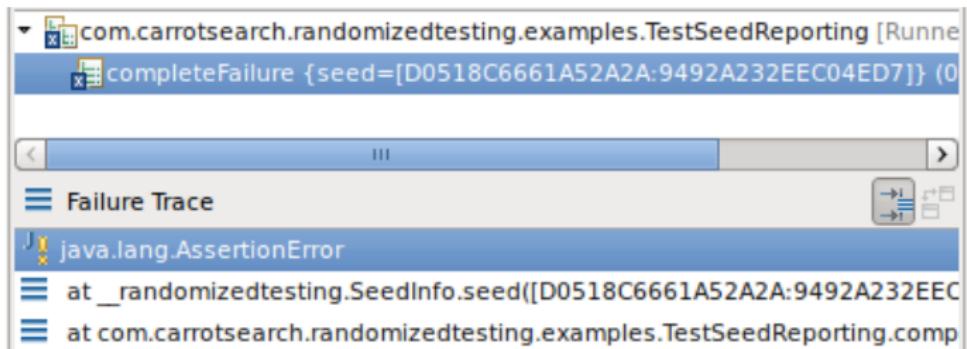
Thread-bound. Defined per lifecycle.
(suite-level, method-level).

```
RandomizedContext ctx =  
    RandomizedContext.current();  
ctx.getTargetClass()  
ctx.getRandom() // => no setSeed()  
ctx.getRandomness().getSeed()  
ctx.isNightly()  
...
```

Seed reporting.

Test method names. Augmented stacks.
Augmented thread names.

```
Assert.assertTrue(  
    RandomizedContext.current()  
        .getRandom().nextBoolean());
```



```
> jstack 6502

...
"TEST-TestScope-org.apache.lucene.util.junitcompat.TestHanging.myTest
-seed#[96F549A92265CBF8]" prio=5 tid=7fd6478b8800 nid=0x1169f1000
waiting on condition [1169ef000]
  java.lang.Thread.State: TIMED_WAITING (sleeping)
at java.lang.Thread.sleep(Native Method)
at org.apache.lucene.util.junitcompat.TestHanging.myTest(TestHanging.java:26)
...
...
```

Shortcuts

extends RandomizedTest

Lots of static utility scaffolding.

Static, but not shared randomness!

- ◆ `S getRandom() : Random`
- ◆ `S randomBoolean() : boolean`
- ◆ `S randomByte() : byte`
- ◆ `S randomShort() : short`

- ◆ `S randomIntBetween(int, int) : int`
- ◆ `S randomFrom(T[]) <T> : T`
- ◆ `S randomFrom(List<T>) <T> : T`

- ◆ `S newTempDir() : File`
- ◆ `S newTempFile() : File`

- ◆ `S randomLocale() : Locale`
- ◆ `S randomTimeZone() : TimeZone`

- ◆ `S randomAsciiString() : String`
- ◆ `S randomCharString(char, char, int) : String`
- ◆ `S randomUnicodeString() : String`
- ◆ `S randomUnicodeString(int) : String`

- `S assumeTrue(boolean, String) : void`
- `S assumeFalse(boolean, String) : void`
- `S assumeNoException(String, Throwable) :`

Shortcuts

extends RandomizedTest

Lots of static utility scaffolding.
Static, but not shared randomness!

```
public class TestSeedReporting2
/* ==> */ extends RandomizedTest {
@Test
public void completeFailure() {
    Assert.assertTrue(randomBoolean());
}
}
```

- ◆ `S getRandom() : Random`
- ◆ `S randomBoolean() : boolean`
- ◆ `S randomByte() : byte`
- ◆ `S randomShort() : short`

- ◆ `S randomIntBetween(int, int) : int`
- ◆ `S randomFrom(T[]) <T> : T`
- ◆ `S randomFrom(List<T>) <T> : T`

- ◆ `S newTempDir() : File`
- ◆ `S newTempFile() : File`

- ◆ `S randomLocale() : Locale`
- ◆ `S randomTimeZone() : TimeZone`

- ◆ `S randomAsciiString() : String`
- ◆ `S randomCharString(char, char, int) : String`
- ◆ `S randomUnicodeString() : String`
- ◆ `S randomUnicodeString(int) : String`

- `S assumeTrue(boolean, String) : void`
- `S assumeFalse(boolean, String) : void`
- `S assumeNoException(String, Throwable) :`

Once you hit a failure...

```
@Seed("deadbeef")
@Repeat(iterations = 10, useConstantSeed = true)
@Test
public void completeFailure2() {
    Assert.assertTrue(randomBoolean());
}
```

Once you hit a failure...

```
@Seed("deadbeef")
@Repeat(iterations = 10, useConstantSeed = true)
@Test
public void completeFailure2() {
    Assert.assertTrue(randomBoolean());
}
```

...or override globally...

```
-Dtests.seed=0:deadbeef // seed format: master[:method]
-Dtests.iter=10
-Dtests.method=completeFailure2
```

Once you hit a failure...

...and see if your test is predictable for that seed.

```
@Seed("deadbeef")
@Repeat(iterations = 10, useConstantSeed = true)
@Test
public void completeFailure2() {
    Assert.assertTrue(randomBoolean());
}
```

▼ completeFailure2 (0.038 s)

- completeFailure2 {#0 seed=[E6AFC4EF6D09A85A:DEADBEEF]}
- completeFailure2 {#1 seed=[E6AFC4EF6D09A85A:DEADBEEF]}
- completeFailure2 {#2 seed=[E6AFC4EF6D09A85A:DEADBEEF]}
- completeFailure2 {#3 seed=[E6AFC4EF6D09A85A:DEADBEEF]}
- completeFailure2 {#4 seed=[E6AFC4EF6D09A85A:DEADBEEF]}
- completeFailure2 {#5 seed=[E6AFC4EF6D09A85A:DEADBEEF]}
- completeFailure2 {#6 seed=[E6AFC4EF6D09A85A:DEADBEEF]}
- completeFailure2 {#7 seed=[E6AFC4EF6D09A85A:DEADBEEF]}
- completeFailure2 {#8 seed=[E6AFC4EF6D09A85A:DEADBEEF]}
- completeFailure2 {#9 seed=[E6AFC4EF6D09A85A:DEADBEEF]}

Once you hit a failure...

...and see if your test is predictable for that seed.

```
@Seed("deadbeef")
@Repeat(iterations = 10, useConstantSeed = false)
@Test
public void completeFailure3() {
    Assert.assertTrue(randomBoolean());
}
```

completeFailure3 (0.030 s)

- completeFailure3 {#0 seed=[13F8CAAE0B90CE99:DEADBEEF]} (0.003 s)
- completeFailure3 {#1 seed=[13F8CAAE0B90CE99:B456BCFCEA6F75C3]}
- completeFailure3 {#2 seed=[13F8CAAE0B90CE99:3ABF2A20BBAB3D08]}
- completeFailure3 {#3 seed=[13F8CAAE0B90CE99:B5181C5D7556621]}
- completeFailure3 {#4 seed=[13F8CAAE0B90CE99:47900468765DA69A]}
- completeFailure3 {#5 seed=[13F8CAAE0B90CE99:D66AD7370BE1EB9A]}
- completeFailure3 {#6 seed=[13F8CAAE0B90CE99:E8B4B3B119D1FB9C]}
- completeFailure3 {#7 seed=[13F8CAAE0B90CE99:740729CB3AC56F32]}
- completeFailure3 {#8 seed=[13F8CAAE0B90CE99:46ABC54D0E7868]}
- completeFailure3 {#9 seed=[13F8CAAE0B90CE99:91209A1F29594F3A]}

Slowly adding regression guards...

**...(note the
random one)...**

```
@Seeds({
    @Seed("deadbeef"),
    @Seed("cafebabe"),
    @Seed("deadbaba"),
    @Seed("baadfood"),
    @Seed()
})
@Test
public void completeFailure4() {
    Assert.assertTrue(randomBoolean());
}
```

Slowly adding regression guards...

...**(note the random one)...**

```
@Seeds({
    @Seed("deadbeef"),
    @Seed("cafebabe"),
    @Seed("deadbaba"),
    @Seed("baadfood"),
    @Seed()
})
@Test
public void completeFailure4() {
    Assert.assertTrue(randomBoolean());
}
```

Test Case	Seed	Time
completeFailure4	(0.018 s)	(0.018 s)
completeFailure4	{seed=[E8AB4A0EA7CA4A0:DEADBEEF]}	(0.004 s)
completeFailure4	{seed=[E8AB4A0EA7CA4A0:A0FABAEB92DB04ED]}	
completeFailure4	{seed=[E8AB4A0EA7CA4A0:CAFEBABE]}	(0.003 s)
completeFailure4	{seed=[E8AB4A0EA7CA4A0:DEADBABA]}	(0.003 s)
completeFailure4	{seed=[E8AB4A0EA7CA4A0:BAADF00D]}	(0.004 s)

@RR: Intermediate

@Nightly

Per test or
class.

```
@Test @Nightly
public void verySlowOne() {
    // very slow test executed
    // on a CI server.
}
```

▼  com.carrotsearch.randomizedtesting.examples.TestNightly [Runne
 verySlowOne {seed=[7E3032F800128375:514B133C1B61211D

@Nightly

Per test or
class.

```
@Test @Nightly
public void verySlowOne() {
    // very slow test executed
    // on a CI server.
}
```

-  com.carrotsearch.randomizedtesting.examples.TestNightly [Runne
  verySlowOne {seed=[7E3032F800128375:514B133C1B61211D

Enabling
nightly tests

```
-Dtests.nightly=true
```

-  com.carrotsearch.randomizedtesting.examples.TestNightly [Runne
  verySlowOne {seed=[D5579D93A8A5C146:FA2CBC57B3D6632E

@TestGroup

Meta groups.

```
@Target({ElementType.ANNOTATION_TYPE})
@Inherited
public @interface TestGroup {
    /** Name of this test group... */
    String name() default "";

    /** Sysproperty to enable/ disable a group... */
    String sysProperty() default "";

    /** Default state... */
    boolean enabled() default true;
}
```

@TestGroup

Meta groups.

```
@Target({ElementType.ANNOTATION_TYPE})
@Inherited
public @interface TestGroup {
    /** Name of this test group... */
    String name() default "";

    /** Sysproperty to enable/ disable a group... */
    String sysProperty() default "";

    /** Default state... */
    boolean enabled() default true;
}
```

@Nightly definition

```
@TestGroup(enabled = false)
public @interface Nightly {
    /** Additional description, if needed. */
    String value() default "";
}
```

@TestGroup

Meta groups.

```
@Target({ElementType.ANNOTATION_TYPE})
@Inherited
public @interface TestGroup {
    /** Name of this test group... */
    String name() default "";

    /** Sysproperty to enable/ disable a group... */
    String sysProperty() default "";

    /** Default state... */
    boolean enabled() default true;
}
```

@Nightly definition

```
@TestGroup(enabled = false)
public @interface Nightly {
    /** Additional description, if needed. */
    String value() default "";
}
```

flexible
IDE searchable

```
@Test
@Nightly @LinuxOnly @RequiresDisplay
public void method() {
    // ...
}
```

@ParametersFactory

For constructors.

```
public class TestParameters extends RandomizedTest {  
    private String name;  
    private int age;  
  
    public TestParameters(  
        @Name("age") int age,  
        @Name("name") String name) {  
        this.name = name; this.age = age;  
    }  
  
    @Test  
    public void testPerson() {  
        assertTrue("Won't like: " + name,  
            age >= 18 && age <= 21);  
    }  
  
    @ParametersFactory  
    public static Iterable<Object[]> parameters() {  
        return Arrays.asList($$  
            $(18, "Dolly"),  
            $(25, "Barbie")));  
    }  
}
```

@ParametersFactory

For constructors.

```
public class TestParameters extends RandomizedTest {  
    private String name;  
    private int age;  
  
    public TestParameters(  
        @Name("age") int age,  
        @Name("name") String name) {  
        this.name = name; this.age = age;  
    }  
  
    @Test  
    public void testPerson() {  
        assertTrue("Won't like: " + name,  
            age >= 18 && age <= 21);  
    }  
  
    @ParametersFactory  
    public static Iterable<Object[]> parameters() {  
        return Arrays.asList($$  
            $(18, "Dolly"),  
            $(25, "Barbie")));  
    }  
}
```

Seed unused, but must
be there.

-  testPerson (0.021 s)
 -  testPerson {age=18 name=Dolly seed=[BDB15CB9B3F203F2
 -  testPerson {age=25 name=Barbie seed=[BDB15CB9B3F203F

Threads and Timeouts

ThreadGroup per suite.

Sub-threads inherit their own randomness (context).

No races—fixed seed (master) for each thread. Repeatable?

Threads and Timeouts

ThreadGroup per suite.

Sub-threads inherit their own randomness (context).

No races—fixed seed (master) for each thread. Repeatable?

@ThreadLeakstm

Left-behind threads failure control. Lingering control.

Threads and Timeouts

ThreadGroup per suite.

Sub-threads inherit their own randomness (context).

No races—fixed seed (master) for each thread. Repeatable?

@ThreadLeakstm

Left-behind threads failure control. Lingering control.

Global and method @Timeout.

Violators cause test or suite failure.

Threads and Timeouts

ThreadGroup per suite.

Sub-threads inherit their own randomness (context).

No races—fixed seed (master) for each thread. Repeatable?

@ThreadLeakstm

Left-behind threads failure control. Lingering control.

Global and method @Timeout.

Violators cause test or suite failure.

Thread termination.

Rationale: no interference from left-behinds.

Interrupt-pause-stop-pause-continue cycle. Problems with Thread.stop().

```
@Test @Timeout(millis = 1000)
public void interruptable() throws Exception {
    Thread.sleep(100000);
}
```

```
Oct 14, 2011 4:49:51 PM randomizedtesting.RandomizedRunner tryToTerminate
WARNING: Attempting to stop thread: Thread-4(#1387438808), currently at:
java.lang.Thread.sleep(Native Method)
randomizedtesting.examples.TestExample5.interruptable(TestExample5.java:12)
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
java.lang.reflect.Method.invoke(Method.java:597)
randomizedtesting.RandomizedRunner.invoke(RandomizedRunner.java:957)
randomizedtesting.RandomizedRunner.access$4(RandomizedRunner.java:943)
randomizedtesting.RandomizedRunner$4.evaluate(RandomizedRunner.java:729)
randomizedtesting.RandomizedRunner.runWithRules(RandomizedRunner.java:733)
randomizedtesting.RandomizedRunner.runSingleTest(RandomizedRunner.java:673)
randomizedtesting.RandomizedRunner.access$3(RandomizedRunner.java:654)
randomizedtesting.RandomizedRunner$3.run(RandomizedRunner.java:380)
java.lang.Thread.run(Thread.java:619)
```

```
Oct 14, 2011 4:49:51 PM randomizedtesting.RandomizedRunner tryToTerminate
WARNING: Interrupted a runaway thread: Thread-4(#1387438808)
```

```
@Test @Timeout(millis = 1000)
public void uninterruptable() throws Exception {
    while (true) /* spin */;
}
```

```
Oct 14, 2011 5:13:58 PM randomizedtesting.RandomizedRunner tryToTerminate
WARNING: Attempting to stop thread: Thread-3(#1753620260), currently at:
randomizedtesting.examples.TestExample5.uninterruptable(TestExample5.java:18)
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
java.lang.reflect.Method.invoke(Method.java:597)
randomizedtesting.RandomizedRunner.invoke(RandomizedRunner.java:961)
randomizedtesting.RandomizedRunner.access$4(RandomizedRunner.java:947)
randomizedtesting.RandomizedRunner$4.evaluate(RandomizedRunner.java:733)
randomizedtesting.RandomizedRunner.runWithRules(RandomizedRunner.java:737)
randomizedtesting.RandomizedRunner.runSingleTest(RandomizedRunner.java:677)
randomizedtesting.RandomizedRunner.access$3(RandomizedRunner.java:658)
randomizedtesting.RandomizedRunner$3.run(RandomizedRunner.java:380)
java.lang.Thread.run(Thread.java:619)
```

```
Oct 14, 2011 5:14:09 PM randomizedtesting.RandomizedRunner tryToTerminate
WARNING: Does not respond to interrupt(), trying to stop(): Thread-3(#1753620260)
Oct 14, 2011 5:14:09 PM randomizedtesting.RandomizedRunner tryToTerminate
WARNING: Stopped a runaway thread: Thread-3(#1753620260)
```

```
@Test @Timeout(millis = 1000)
public void christopherLambert() throws Exception {
    while (true) {
        try { Thread.sleep(1000); } catch (Throwable t) {}
    }
}
```

Oct 14, 2011 5:13:35 PM randomizedtesting.RandomizedRunner tryToTerminate
WARNING: Attempting to stop thread: Thread-2(#2018812712), currently at:
java.lang.Thread.sleep(Native Method)
randomizedtesting.examples.TestExample5.christopherLambert(TestExample5.java:25)
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
java.lang.reflect.Method.invoke(Method.java:597)
randomizedtesting.RandomizedRunner.invoke(RandomizedRunner.java:961)
randomizedtesting.RandomizedRunner.access\$4(RandomizedRunner.java:947)
randomizedtesting.RandomizedRunner\$4.evaluate(RandomizedRunner.java:733)
randomizedtesting.RandomizedRunner.runWithRules(RandomizedRunner.java:737)
randomizedtesting.RandomizedRunner.runSingleTest(RandomizedRunner.java:677)
randomizedtesting.RandomizedRunner.access\$3(RandomizedRunner.java:658)
randomizedtesting.RandomizedRunner\$3.run(RandomizedRunner.java:380)
java.lang.Thread.run(Thread.java:619)

Oct 14, 2011 5:13:46 PM randomizedtesting.RandomizedRunner tryToTerminate
WARNING: Does not respond to interrupt(), trying to stop(): Thread-2(#2018812712)
Oct 14, 2011 5:13:57 PM randomizedtesting.RandomizedRunner tryToTerminate
SEVERE: Could not interrupt or stop thread: Thread-2(#2018812712)

finished after 50002 seconds

Runs: 4/4

Errors: 5

Failures: 0

- ▼ com.carrotsearch.randomizedtesting.examples.barcelona.TestExample5 [Runner: JUnit 4] (36.609 s)
- ✗ interruptable [2DFCBB32185BBAFF:3DF29DF97F4E159] (1.041 s)
 - ✗ christopherLambert [2DFCBB32185BBAFF:7DD8831A79157970] (23.021 s)
 - ✗ leftOverSubThread [2DFCBB32185BBAFF:F7B60EEA39BA1B08] (0.513 s)
 - ✗ uninterruptable [2DFCBB32185BBAFF:99A70041551F4025] (12.032 s)

Failure Trace

```
java.lang.RuntimeException: Test case thread timed out (and NOT TERMINATED, left in state TIMED_WAITING): Thread[T
    at java.lang.Thread.sleep(Native Method)
    at com.carrotsearch.randomizedtesting.examples.barcelona.TestExample5.christopherLambert(TestExample5.java:27)
    at com.carrotsearch.randomizedtesting.RandomizedRunner.invoke(RandomizedRunner.java:978)
    at com.carrotsearch.randomizedtesting.RandomizedRunner.access$6(RandomizedRunner.java:964)
```

@RR: Advanced

@ThreadLeaks

Background
threads from
nowhere.

```
@Test @Repeat(iterations = 10)
public void leakInExecutors() throws Exception {
    ExecutorService exec =
        Executors.newCachedThreadPool();
    for (int i = 0; i < 5; i++) {
        exec.submit(new Runnable() {
            public void run() {
                sleep(100);
            }
        });
    }

    // "Orderly" shutdown. Wait for executing tasks.
    exec.shutdown();
    exec.awaitTermination(5, TimeUnit.SECONDS);
    assertTrue(exec.isShutdown());
    assertTrue(exec.isTerminated());
}
```

Runs: 10/10

Errors: 2

Failures: 0

- ▼ com.carrotsearch.randomizedtesting.examples.TestThreadLeaks [Runner: JUnit 4] (1.280 s)
 - ▼ leakInExecutors (1.280 s)
 - leakInExecutors {#0 seed=[7E7F79C0262BBA19:2DEB34FB17D574EF]} (0.221 s)
 - leakInExecutors {#1 seed=[7E7F79C0262BBA19:99BD88072317BFC3]} (0.104 s)
 - leakInExecutors {#2 seed=[7E7F79C0262BBA19:17541EDB72D3F708]} (0.114 s)
 - leakInExecutors {#3 seed=[7E7F79C0262BBA19:26BAB53E1E2DAC21]} (0.109 s)
 - leakInExecutors {#4 seed=[7E7F79C0262BBA19:6A7B3093BF256C9A]} (0.106 s)
 - leakInExecutors {#5 seed=[7E7F79C0262BBA19:FB81E3CCC299219A]} (0.198 s)
 - leakInExecutors {#6 seed=[7E7F79C0262BBA19:C55F874AD0A9319C]} (0.105 s)
 - leakInExecutors {#7 seed=[7E7F79C0262BBA19:59EC1D30F3BDA532]} (0.106 s)
 - leakInExecutors {#8 seed=[7E7F79C0262BBA19:6B40F85E8476B268]} (0.107 s)
 - leakInExecutors {#9 seed=[7E7F79C0262BBA19:BCCBAEE4E021853A]} (0.108 s)

Failure Trace

```
J com.carrotsearch.randomizedtesting.ThreadingError: Left-over thread detected (and term
  at __randomizedtesting.SeedInfo.seed([7E7F79C0262BBA19:2DEB34FB17D574EF]:0)
  at sun.misc.Unsafe.unpark(Native Method)
  at java.util.concurrent.locks.LockSupport.unpark(LockSupport.java:124)
  at java.util.concurrent.locks.AbstractQueuedSynchronizer.unparkSuccessor(AbstractQueu
  at java.util.concurrent.locks.AbstractQueuedSynchronizer.release(AbstractQueuedSynchr
  at java.util.concurrent.locks.ReentrantLock.unlock(ReentrantLock.java:431)
  at java.util.concurrent.ThreadPoolExecutor.workerDone(ThreadPoolExecutor.java:1013)
  at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:912)
  at java.lang.Thread.run(Thread.java:619)
```

@ThreadLeaks

Linger a bit
(max time!)

```
@Test @Repeat(iterations = 10)
@ThreadLeak(linger = 1000)
public void leakInExecutors() throws Exception {
    ...
}
```

@ThreadLeaks and stop()

@ThreadLeaks and stop()

**After a thread is stopped its resources should not be reused
(may be left in an inconsistent state!).**

Applies to executors and thread pools in particular.

@Validators, @Listeners,...

```
@RunWith(RandomizedRunner.class)
@TestMethodProviders({
    LuceneJUnit3MethodProvider.class,
    JUnit4MethodProvider.class
})
@Validators({
    RequireAssertions.class,
    NoStaticHooksShadowing.class,
    NoInstanceHooksOverrides.class
})
@Listeners({
    PrintReproduceInfoListener.class
})
@SeedDecorators({MixWithSuiteName.class}) // See LUCENE-3995 for rationale.
@ThreadLeaks(failTestIfLeaking = false)
public abstract class LuceneTestCase extends Assert {
    ...
}
```

...more at:

<https://github.com/carrotsearch/randomizedtesting/issues/>

<JUnit4>

Parallel <junit> for Ant

Ant task and Maven plugin

Needs to be loaded.

Forks multiple JVMs

And runs tests in parallel. Load balances.

Coordinated listeners

Aggregated event stream.

```
<taskdef resource="com/carrotsearch/junit4/antlib.xml">
<classpath>
  <fileset dir="${common.dir}/test-framework/lib">
    <include name="junit4-ant-*.jar" />
    <include name="junit-*.jar" />
  </fileset>
</classpath>
</taskdef>
```

```
<taskdef resource="com/carrotsearch/junit4/antlib.xml">
<classpath>
  <fileset dir="${common.dir}/test-framework/lib">
    <include name="junit4-ant-*.jar" />
    <include name="junit-*.jar" />
  </fileset>
</classpath>
</taskdef>
```

```
<junit4 parallelism="auto" seed="..." ...>
  ...
</junit4>
```

**Attributes compatible with standard <junit>.
Close integration with RandomizedRunner.**

```
[junit4] <JUnit4> says hello! Master seed: 4AD7B1A85EF95F35
[junit4] Expected execution time on JVM J0: 8.81s
[junit4] Expected execution time on JVM J1: 8.81s
[junit4] Expected execution time on JVM J2: 8.81s
[junit4] Executing 114 suites with 3 JVMs.
[junit4]
[junit4] Suite: org.carrot2.util.attribute.constraint.IsDirectoryConstraintTest
[junit4] Completed on J2 in 0.21s, 8 tests
[junit4]
[junit4] Suite: org.carrot2.util.attribute.BindableDescriptorBuilderTest
[junit4] Completed on J1 in 2.63s, 8 tests
[junit4]
[junit4] Suite: org.carrot2.clustering.stc.STCCLusteringAlgorithmTest
[junit4] IGNOR/A 0.00s J0 | STCCLusteringAlgorithmTest.testStress
[junit4] > Assumption #1: 'nightly' test group is disabled (@Nightly)
[junit4] Completed on J0 in 0.22s, 9 tests, 1 skipped
...
[junit4] JVM J0: 1.15 .. 24.76 = 23.60s
[junit4] JVM J1: 1.19 .. 23.45 = 22.26s
[junit4] JVM J2: 1.30 .. 23.48 = 22.18s
[junit4] Execution time total: 24 seconds
[junit4] Tests summary: 114 suites, 1095 tests, 1 error, 116 ignored (111 assumptions)
```

carrot2.master: 1 test had errors

1095 tests executed in 51405 ms on 3 slaves.
1 error, 116 ignored, 978 passed

package, class, method name (Alt+Shift+S to focus)

show: pass ignored error fail

view: packages classes
methods console

Method ↑

Result ↓ JVM Start Time [ms]

org.carrot2.util.attribute.AttributeTransformerFromStringTest.testFileResourceFile com.carrotsearch.randomizedtesting.ThreadingError: Left-over thread detected (and terminated): Thread[Poller SunPKCS11-Darwin,1,] (stack trace is a snapshot location of the thread at the moment of killing, see the system logger for probes and more information). at __randomizedtesting.SeedInfo.seed([5774B06DC8249]:0)	ERROR	1	11:03:18.258	950
org.carrot2.core.ControllerTest\$CachingControllerCachingOnCachingTests.testConcurr...	IGNORED	0	11:03:35.198	0
org.carrot2.core.ControllerTest\$CachingPoolingControllerCachingOnCachingTests.test...	IGNORED	0	11:03:31.565	0
org.carrot2.dcs.DcsAppTest.testDcsConfigLocation	IGNORED	2	11:03:33.348	0
org.carrot2.util.CharArrayUtilsTest.bufferTooSmall	IGNORED	0	11:03:13.190	0
org.carrot2.util.CharArrayUtilsTest.wordTooShort	IGNORED	0	11:03:13.205	0
org.carrot2.clustering.kmeans.BisectingKMeansClusteringAlgorithmTest.testStress com.carrotsearch.randomizedtesting.InternalAssumptionViolatedException: 'nightly' test group is disabled (@Nightly) ***	IGNORED	1	11:03:34.147	4

carrot2.master: 1 test had errors

9 tests executed in 1839 ms on 2 slaves.
1 error, 8 passed

show: pass ignored error fail without output view: packages classes methods console

org.carrot2.util.attribute.AttributeTransformerFromStringTest
2012-05-14 11:03:18,223 WARN org.carrot2.util.ReflectionUtils: Could not load class:
xorg.carrot2.util.attribute.AttributeTransformerFromStringTest
(xorg.carrot2.util.attribute.AttributeTransformerFromStringTest).
May 14, 2012 11:03:19 AM com.carrotsearch.randomizedtesting.RandomizedRunner reportStackProbes
WARNING: 9 stack trace probe(s) taken and the constant root was:
...
java.lang.Thread.sleep(Native Method)
[...sun.*]
java.lang.Thread.run(Thread.java:680)

Diverging stack paths from individual probes (if different than the common root):
(all stacks constant.)

May 14, 2012 11:03:19 AM com.carrotsearch.randomizedtesting.RandomizedRunner tryToTerminate
WARNING: Attempting to terminate thread: Poller SunPKCS11-Darwin(#1547065998), currently at:
java.lang.Thread.sleep(Native Method)
[...sun.*]
java.lang.Thread.run(Thread.java:680)

May 14, 2012 11:03:19 AM com.carrotsearch.randomizedtesting.RandomizedRunner tryToTerminate
WARNING: Interrupted a runaway thread: Poller SunPKCS11-Darwin(#1547065998)

org.carrot2.dcs.DcsAppTest
2012-05-14 11:03:32,621 INFO dcs: Starting DCS...
2012-05-14 11:03:33,182 INFO dcs: DCS started on port: 57913
2012-05-14 11:03:33,519 INFO dcs: DCS stopped.

org.carrot2.core.ControllerTest
2012-05-14 11:03:25,239 WARN org.carrot2.util.ReflectionUtils: Could not load class: nonexistent-component (nonexistent-component).
2012-05-14 11:03:26,693 INFO org.carrot2.core.PoolingProcessingComponentManager\$NonPrimitiveInputAttributesCheck: An object of a

testUnknownComponentId
warningOnNonThreadSafeInitInputInstanceAttribut
eProvidedOnInit

Lucene tests speedup.

```
> ant test-core -Dtests.seed=deadbeef -Dtests.jvms=1  
...  
[junit4] JVM J0: 0.43 .. 302.97 = 302.54s  
[junit4] Execution time total: 5 minutes 3 seconds
```

Lucene tests speedup.

```
> ant test-core -Dtests.seed=deadbeef -Dtests.jvms=1  
...  
[junit4] JVM J0: 0.43 .. 302.97 = 302.54s  
[junit4] Execution time total: 5 minutes 3 seconds
```

```
> ant test-core -Dtests.seed=deadbeef -Dtests.jvms=2  
...  
[junit4] JVM J0: 0.44 .. 169.60 = 169.17s  
[junit4] JVM J1: 0.44 .. 169.44 = 169.00s  
[junit4] Execution time total: 2 minutes 49 seconds
```

Lucene tests speedup.

```
> ant test-core -Dtests.seed=deadbeef -Dtests.jvms=1
...
[junit4] JVM J0: 0.43 .. 302.97 = 302.54s
[junit4] Execution time total: 5 minutes 3 seconds
```

```
> ant test-core -Dtests.seed=deadbeef -Dtests.jvms=2
...
[junit4] JVM J0: 0.44 .. 169.60 = 169.17s
[junit4] JVM J1: 0.44 .. 169.44 = 169.00s
[junit4] Execution time total: 2 minutes 49 seconds
```

```
> ant test-core -Dtests.seed=deadbeef -Dtests.jvms=3
...
[junit4] Execution time total: 2 minutes 13 seconds
```

Lucene tests speedup.

```
> ant test-core -Dtests.seed=deadbeef -Dtests.jvms=1
...
[junit4] JVM J0: 0.43 .. 302.97 = 302.54s
[junit4] Execution time total: 5 minutes 3 seconds
```

```
> ant test-core -Dtests.seed=deadbeef -Dtests.jvms=2
...
[junit4] JVM J0: 0.44 .. 169.60 = 169.17s
[junit4] JVM J1: 0.44 .. 169.44 = 169.00s
[junit4] Execution time total: 2 minutes 49 seconds
```

```
> ant test-core -Dtests.seed=deadbeef -Dtests.jvms=3
...
[junit4] Execution time total: 2 minutes 13 seconds
```

```
> ant test-core -Dtests.seed=deadbeef -Dtests.jvms=4
...
JVM J0: 0.70 .. 112.69 = 111.99s
JVM J1: 0.69 .. 112.82 = 112.12s
JVM J2: 0.70 .. 127.57 = 126.87s
JVM J3: 0.69 .. 118.07 = 117.38s
Execution time total: 2 minutes 7 seconds
```

Summary

Randomized Testing

Complex boundary conditions.

May or may not hit them, but there is a chance!

Input noise resilience.

You simply cannot predict what will appear on input.

Unexpected component-component interactions.

Pairwise component compatibility.

Unexpected environment interactions.

JVM, operating system differences.

Tool support.

Not really crucial (can be handcrafted), but a nice-to-have.



dawid.weiss@carrotsearch.com

Randomized testing package is @labs:
<http://labs.carrotsearch.com/randomizedtesting.html>

Software Failure. Press left mouse button to continue.

Guru Meditation #00000025.65045338