

Modular Apache Spark: Transform Your Code into Pieces

Albert Franzl (@FranziCros), Schibsted Media Group

#SAISDev3



Slides available in:



bit.ly/SparkSummit-afranzi


About me



Modular Apache Spark: Transform Your Code into Pieces



We will learn:

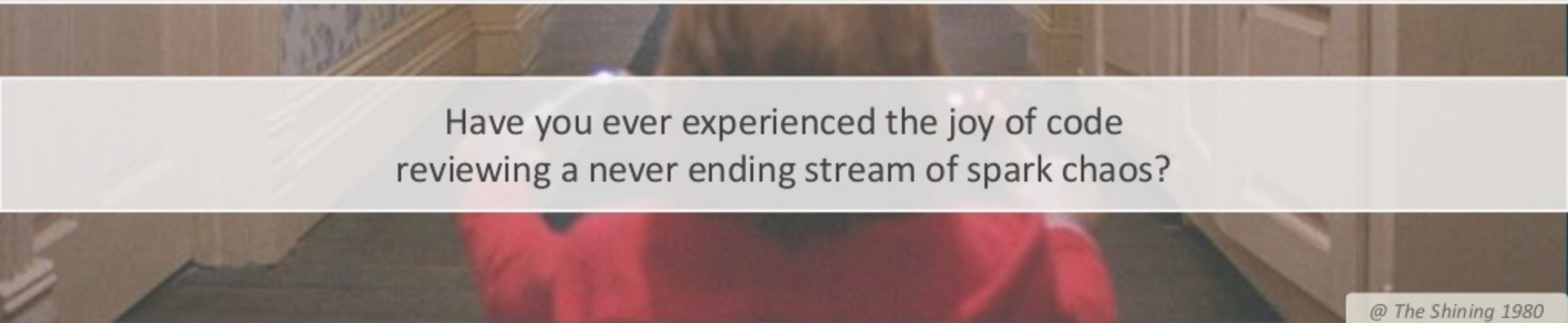
- How we simplified our spark code by modularizing
- How we increased our test coverage in our spark code by using the spark-testing-base provided by Holden Karau
- How we reduced the test time execution by skipping unaffected tests -- > less coffees 

We will learn:

- How we simplified our spark code by modularizing
- How we increased our test coverage in our spark code by using the spark-testing-base provided by Holden Karau
- How we reduced the test time execution by skipping unaffected tests -- > less coffees 



Did you play with duplicated code cross your Spark Jobs?



Have you ever experienced the joy of code reviewing a never ending stream of spark chaos?

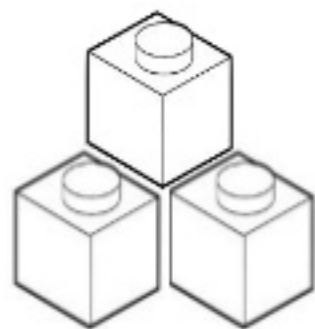
@ The Shining 1980



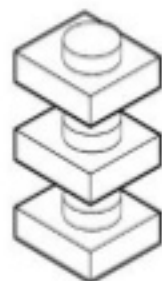
Please! Don't play with duplicated code never ever!

@ The Shining 1980

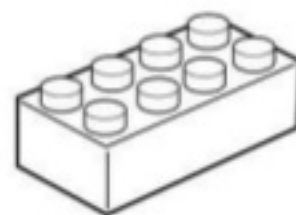
Fragment the Spark Job



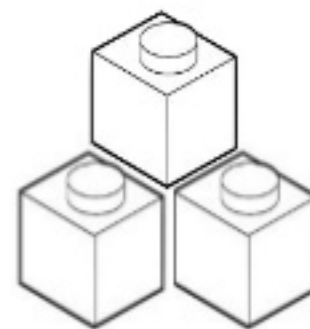
Spark Readers



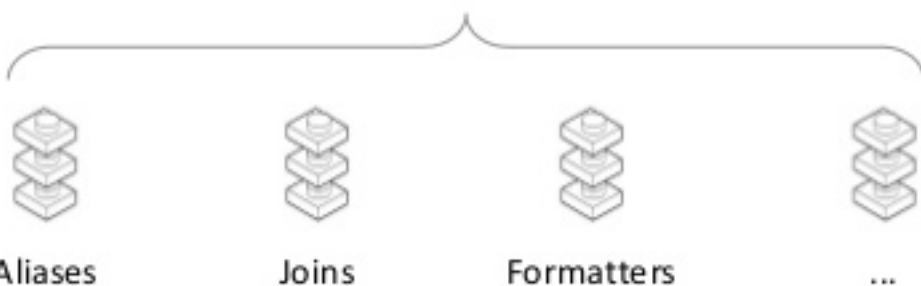
Transformation



Task Context



Spark Writers

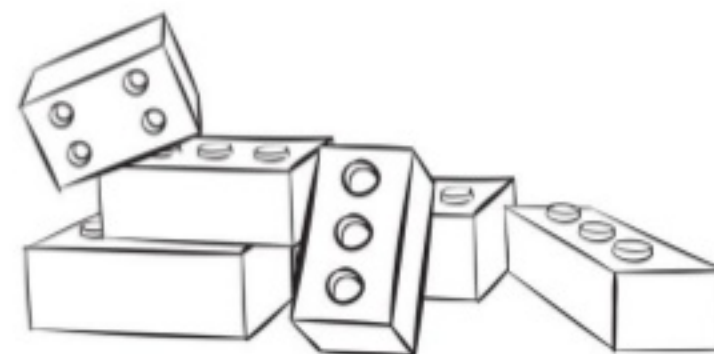


Aliases

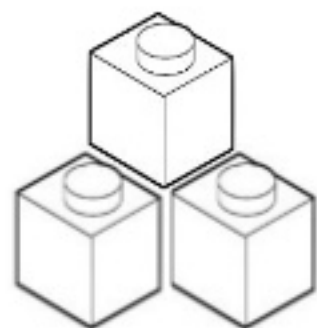
Joins

Formatters

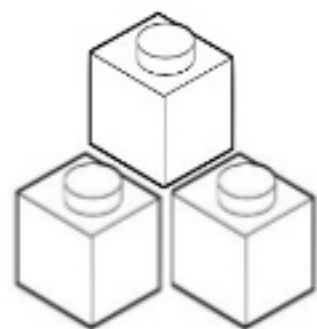
...



Readers / Writers



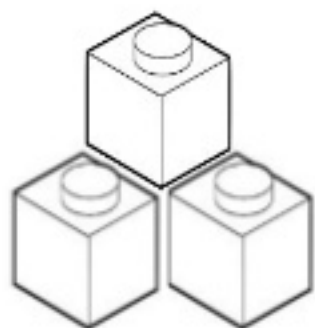
Spark Readers



Spark Writers

- Enforce schemas
- Use schemas to read only the fields you are going to use
- Provide Readers per Dataset & attach its sources to it
- Share schemas & sources between Readers & Writers
- GDPR compliant by design

Readers



Spark Readers

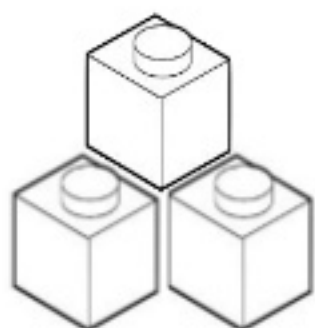
```
val userBehaviourSchema: StructType = ???

val userBehaviourPath =
  Path("s3://<bucket>/user_behaviour/year=2018/month=10/day=03/hour=12/gen=27/")
  GDPR

val userBehaviourReader = ReaderBuilder(PARQUET)
  .withSchema(userBehaviourSchema)
  .withPath(userBehaviourPath)
  .buildReader()

val df: DataFrame = userBehaviourReader.read()
```

Readers



Spark Readers

```
val userBehaviourSchema: StructType = ???
```

```
// Path structure - s3://<bucket>/user_behaviour/[year]/[month]/[day]/[hour]/[gen]/
```

```
val userBehaviourBasePath = Path("s3://<bucket>/user_behaviour/")
```

```
val startDate: ZonedDateTime = ZonedDateTime.now(ZoneOffset.UTC)
```

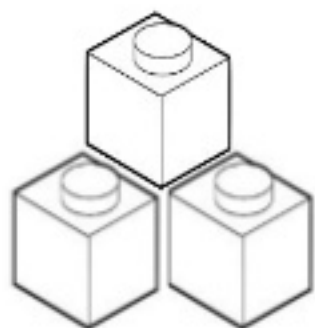
```
val halfDay: Duration = Duration.ofHours(12)
```

```
val userBehaviourPaths: Seq[Path] = PathBuilder  
  .latestGenHourlyPaths(userBehaviourBasePath, startDate, halfDay)
```

```
val userBehaviourReader = ReaderBuilder(PARQUET)  
  .withSchema(userBehaviourSchema)  
  .withPath(userBehaviourPaths: _*)  
  .buildReader()
```

```
val df: DataFrame = userBehaviourReader.read()
```


Readers



Spark Readers

```
val userBehaviourSchema: StructType = ???

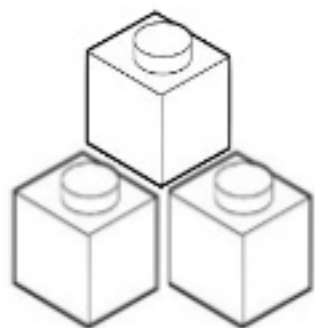
val userBehaviourBasePath = Path("s3://<bucket>/user_behaviour/")

val startDate: ZonedDateTime = ZonedDateTime.now(ZoneOffset.UTC)
val halfDay: Duration = Duration.ofHours(12)

val userBehaviourReader = ReaderBuilder(PARQUET)
    .withSchema(userBehaviourSchema)
    .withHourlyPathBuilder(userBehaviourBasePath, startDate, halfDay)
    .buildReader()

val df: DataFrame = userBehaviourReader.read()
```

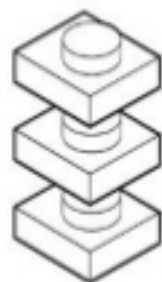
Readers



Spark Readers

```
val df: DataFrame = UserBehaviourReader.read(startDate, halfDay)
```

Transforms

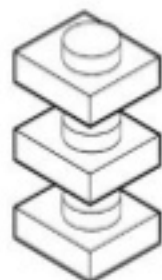


Transformation

```
def transform[U](t: (Dataset[T]) => Dataset[U]): Dataset[U]
```

```
Dataset[T] => magic => Dataset[U]
```

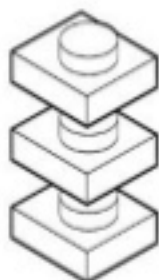
Transforms



Transformation

```
def withGreeting(df: DataFrame): DataFrame = {  
    df.withColumn("greeting", lit("hello world"))  
}  
  
def extractFromJson(colName: String,  
    outputColName: String,  
    jsonSchema: StructType)(df: DataFrame): DataFrame = {  
    df.withColumn(outputColName, from_json(col(colName), jsonSchema))  
}
```

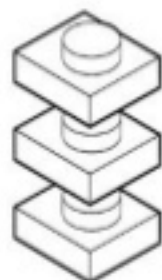

Transforms



Transformation

```
def onlyClassifiedAds(df: DataFrame): DataFrame = {  
  df.filter(col("event_type") === "View")  
    .filter(col("object_type") === "ClassifiedAd")  
}  
  
def dropDuplicates(df: DataFrame): DataFrame = {  
  df.dropDuplicates()  
}  
  
def cleanedCity(df: DataFrame): DataFrame = {  
  df.withColumn("city", getCityUdf(col("object.location.address")))  
}  
  
val cleanupTransformations: Seq[DataFrame => DataFrame] = Seq(  
  dropDuplicates,  
  cleanedCity,  
  onlyClassifiedAds  
)  
  
val df: DataFrame = UserBehaviourReader.read(startDate, halfDay)  
val classifiedAdsDF = df.transforms(cleanupTransformations: _*)
```

Transforms



Transformation

```
val cleanupTransformations: Seq[DataFrame => DataFrame] = Seq(  
    dropDuplicates,  
    cleanedCity,  
    onlyClassifiedAds  
)  
  
val df: DataFrame = UserBehaviourReader.read(startDate, halfDay)  
val classifiedAdsDF = df.transforms(cleanupTransformations: _*)
```

“As a data consumer, I only need to pick up which transformations I would like to apply, instead of coding them from scratch.”

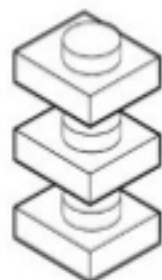
“It’s like cooking, engineers provide manufactured ingredients (transformations) and Data Scientists use the required ones for a successful receipt.”

Transforms - Links of Interest



github.com/MrPowers/spark-daria

"Spark helper methods to maximize developer productivity."




Transformation


"DataFrame transformations can be defined with arguments so they don't make assumptions about the schema of the underlying DataFrame." - by Matthew Powers.

bit.ly/Spark-ChainingTransformations

bit.ly/Spark-SchemaIndependentTransformations

We will learn:

- How we simplified our spark code by modularizing
- How we increased our test coverage in our spark code by using the spark-testing-base provided by Holden Karau
- How we reduced the test time execution by skipping unaffected tests -- > less coffees 



Did you put untested Spark jobs into production?

*"Mars Climate Orbiter destroyed
because of a Metric System Mixup (1999)"*

Testing with Holden Karau



github.com/holdenk/spark-testing-base

"Base classes to use when writing tests with Spark."

- Share Spark Context between tests from the same suite
- Provide methods to make tests easier
 - Fixture Readers
 - Json to DF converters
 - Extra validators



github.com/MrPowers/spark-fast-tests

"An alternative to spark-testing-base to run tests in parallel without restarting Spark Session after each test file."

Testing SharedSparkContext

```
package com.holdenkarau.spark.testing

import java.util.Date

import org.apache.spark._
import org.scalatest.{BeforeAndAfterAll, Suite}

/**
 * Shares a local `SparkContext` between all tests in a suite
 * and closes it at the end. You can share between suites by enabling
 * reuseContextIfPossible.
 */
trait SharedSparkContext extends BeforeAndAfterAll with SparkContextProvider {
  self: Suite =>

  ...

  protected implicit def reuseContextIfPossible: Boolean = false

  ...
}
```


Testing

```
package com.schibsted.insights.test

trait SparkSuite extends DataFrameSuiteBase {
  self: Suite =>

  override def reuseContextIfPossible: Boolean = true

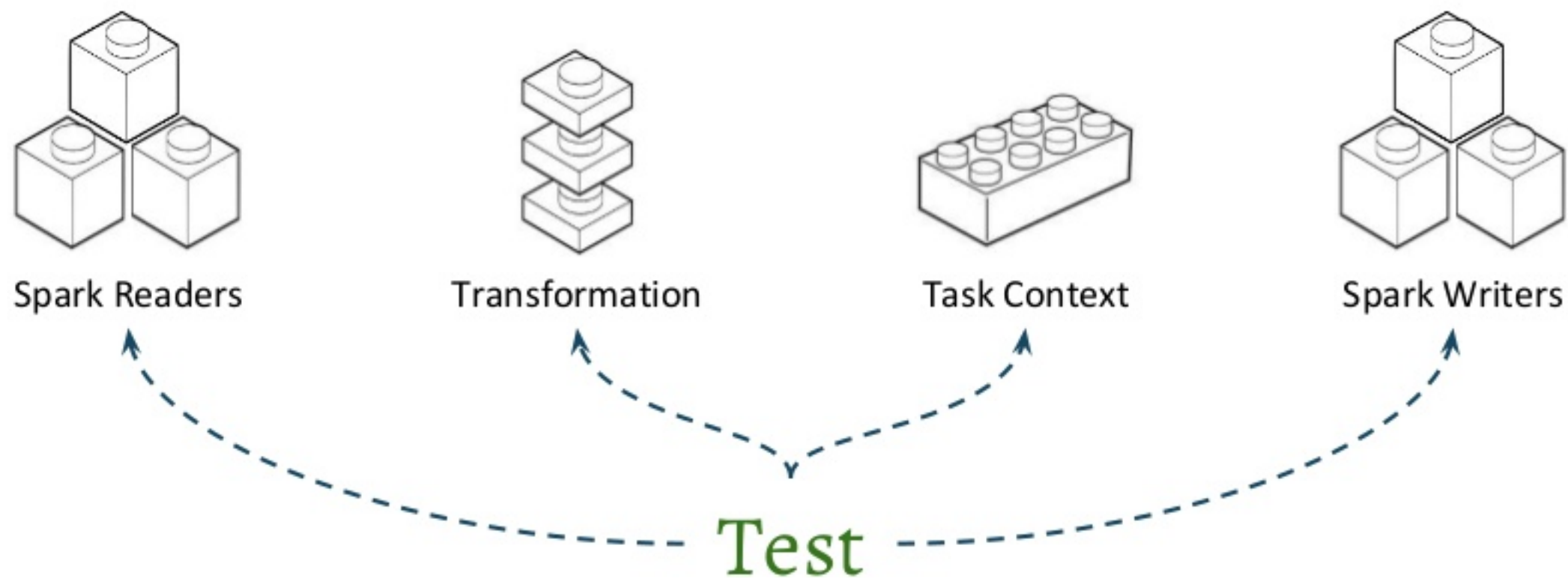
  protected def createDF(data: Seq[Row], schema: StructType): DataFrame = {
    spark.createDataFrame(spark.sparkContext.parallelize(data), schema)
  }

  protected def jsonFixtureToDF(fileName: String, schema: Option[StructType] = None): DataFrame = {
    val fixtureContent = readFixtureContent(fileName)
    val fixtureJson = fixtureContentToJson(fixtureContent)
    jsonToDF(fixtureJson, schema)
  }

  protected def checkSchemas(inputSchema: StructType, expectedSchema: StructType): Unit = {
    assert(inputSchema.fields.sortBy(_.name).deep == expectedSchema.fields.sortBy(_.name).deep)
  }


  ...
}
```



Testing



Testing each piece independently helps testing all together.

We will learn:

- How we simplified our spark code by modularizing
- How we increased our test coverage in our spark code by using the spark-testing-base provided by Holden Karau
- How we reduced the test time execution by skipping unaffected tests -- > less coffees 



Are tests taking too long to execute?



Junit4Git by Raquel Pau



github.com/rpau/junit4git

"JUnit Extensions for Test Impact Analysis."

JUnit4Git

"This is a JUnit extension that ignores those tests that are not related with your last changes in your Git repository."

Junit4Git - Gradle conf

```
configurations {  
    agent  
}  
  
dependencies {  
    testCompile("org.walkmod:scalatest4git_2.11:${version}")  
  
    agent "org.walkmod:junit4git-agent:${version}"  
}  
  
test.doFirst {  
    jvmArgs "-javaagent:${configurations.agent.singleFile}"  
}
```

@RunWith(classOf[ScalaGitRunner])

The diagram consists of a box containing the annotation `@RunWith(classOf[ScalaGitRunner])`. An arrow points from the `agent` dependency line in the Gradle configuration above to this box. Another arrow points from the `testCompile` dependency line to the same box, indicating that both dependencies are used by the `ScalaGitRunner`.

Junit4Git - Travis with Git notes



.travis.yml

```
before_install:
```

```
- echo -e "machine github.com\n  login $GITHUB_TOKEN" >> ~/.netrc
```

```
after_script:
```

```
- git push origin refs/notes/tests:refs/notes/tests
```

Junit4Git - Runners

```
import org.scalatest.junit.JUnitRunner

@RunWith(classOf[JUnitRunner])
abstract class UnitSpec extends FunSuite
```

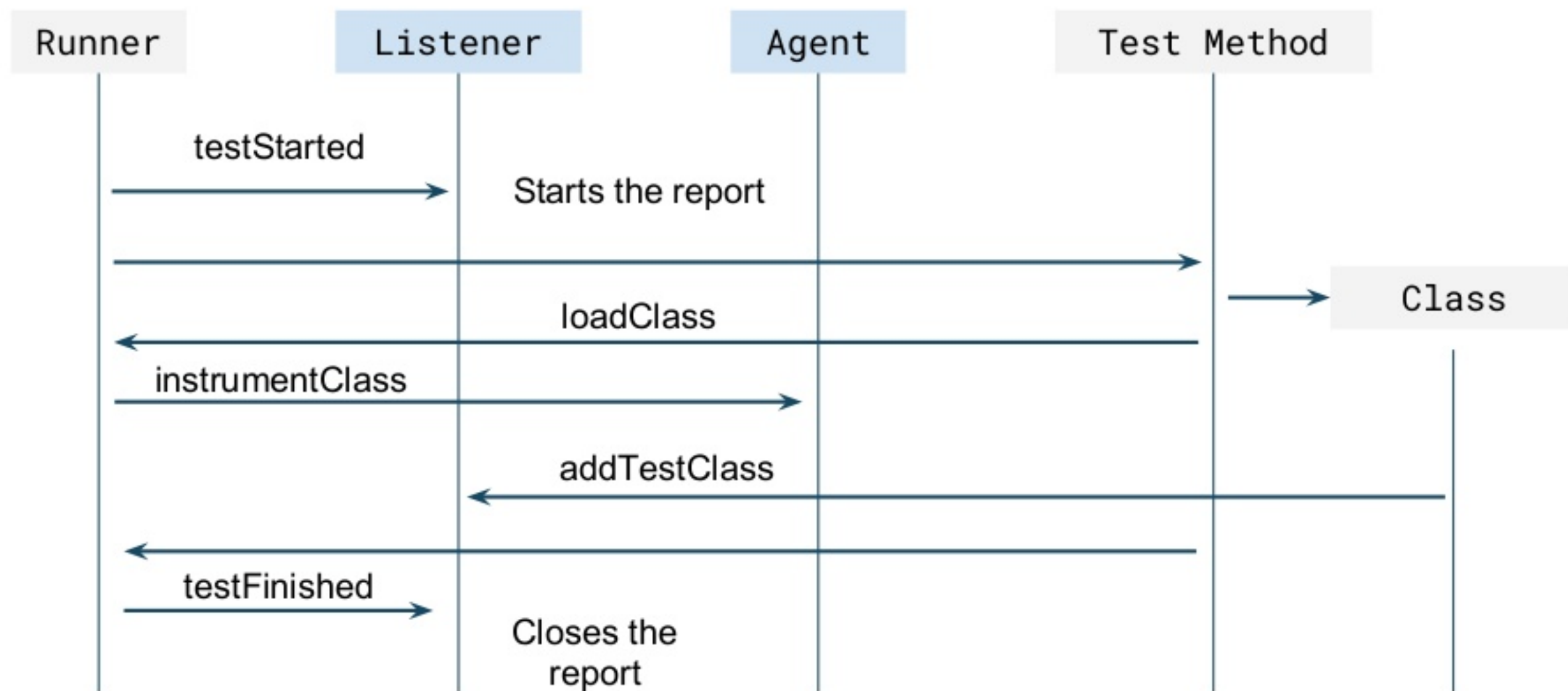
Tests to run always

```
import org.scalatest.junit.ScalaGitRunner

@RunWith(classOf[ScalaGitRunner])
abstract class IntegrationSpec extends FunSuite
```

Tests to run on code changes

Junit4Git



Junit4Git - Notes

```
afranzi:~/Projects/insights-core$ git notes show
```

```
[
  {
    "test": "com.schibsted.insights.core.ContextSuite",
    "method": "Context Classes with SptCommonBucketConf should contain a sptCommonBucket Field",
    "classes": [
      "com.schibsted.insights.core.Context",
      "com.schibsted.insights.core.ContextSuite$$anonfun$1$$anon$1",
      "com.schibsted.insights.core.Context$",
      "com.schibsted.insights.core.SptCommonBucketConf$class",
      "com.schibsted.insights.core.Context$$anonfun$configParamWithLogging$1"
    ]
  }
  ...
]
```

```
com.schibsted.insights.core.ContextSuite > Context Classes with SptCommonBucketConf should
contain a sptCommonBucket Field SKIPPED
```

```
com.schibsted.insights.core.ContextSuite > Context Classes with InsightsBucketConf should
contain a insightsBucket Field SKIPPED
```





Real Impact Testing Analysis For JVM

 bit.ly/SlidesJunit4Git

The Rise of Test Impact Analysis

 bit.ly/TestImpactAnalysis

Summary

-  Use Transforms as pills
-  Don't duplicate code between Spark Jobs
-  Modularize your code
-  Don't be afraid of testing everything



Share all you built as a library, so others can reuse your code.

