Akka TDD实践

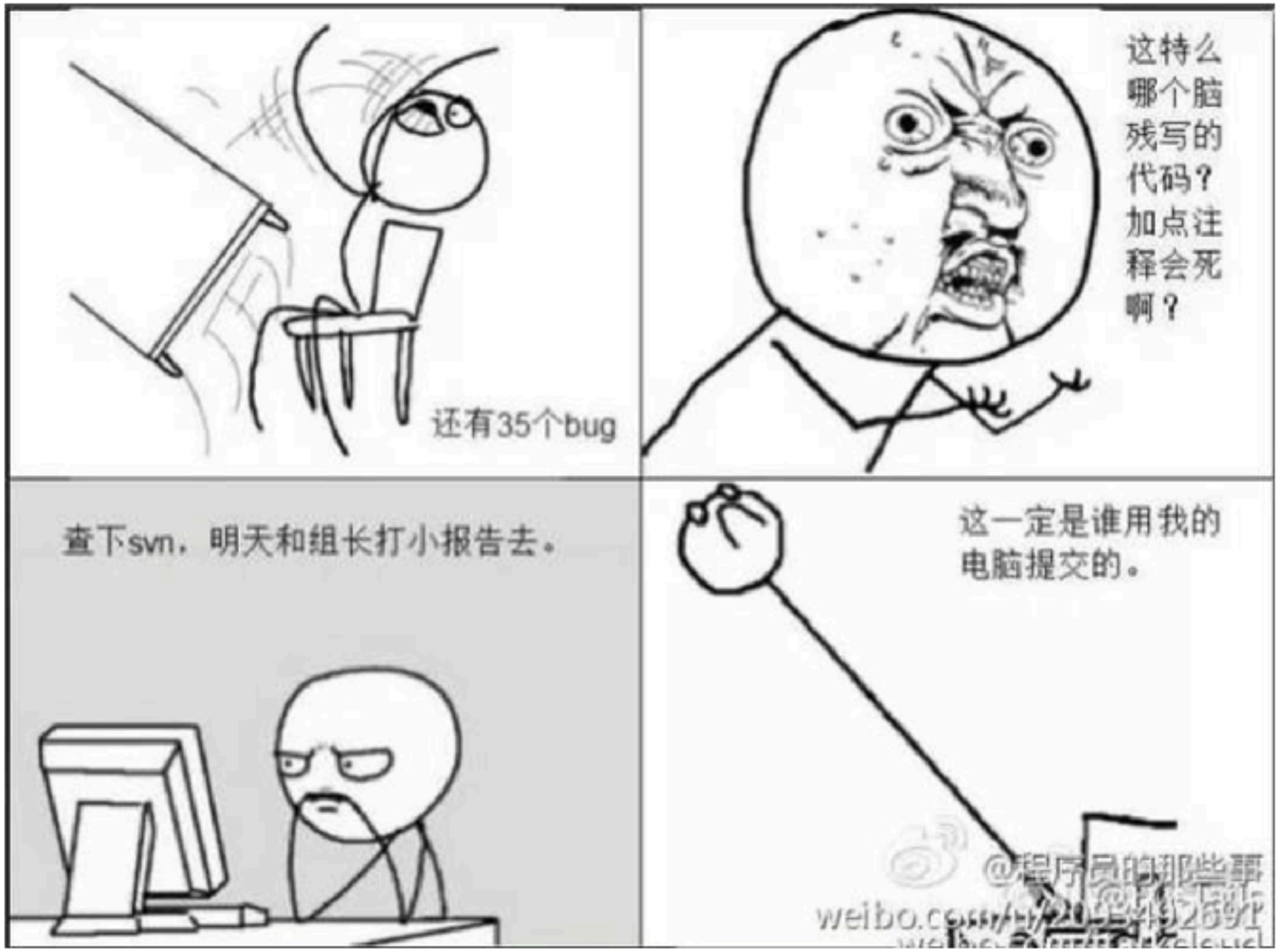
张健

Akka TDD实践

- 什么是TDD?
- Scala Test Framework
- Akka Test Kits
- 演示

“从前,有个这样的故事”

我觉得这个活我能在周末帮你们搞定


还有35个bug
这特么哪个脑残写的代码？加点注释会死啊？
查下svn，明天和组长打小报告去。
这一定是谁用我的电脑提交的。
@程序员的那些事

“ 猝，享年23岁 ”

*TDD*带给了我们什么？

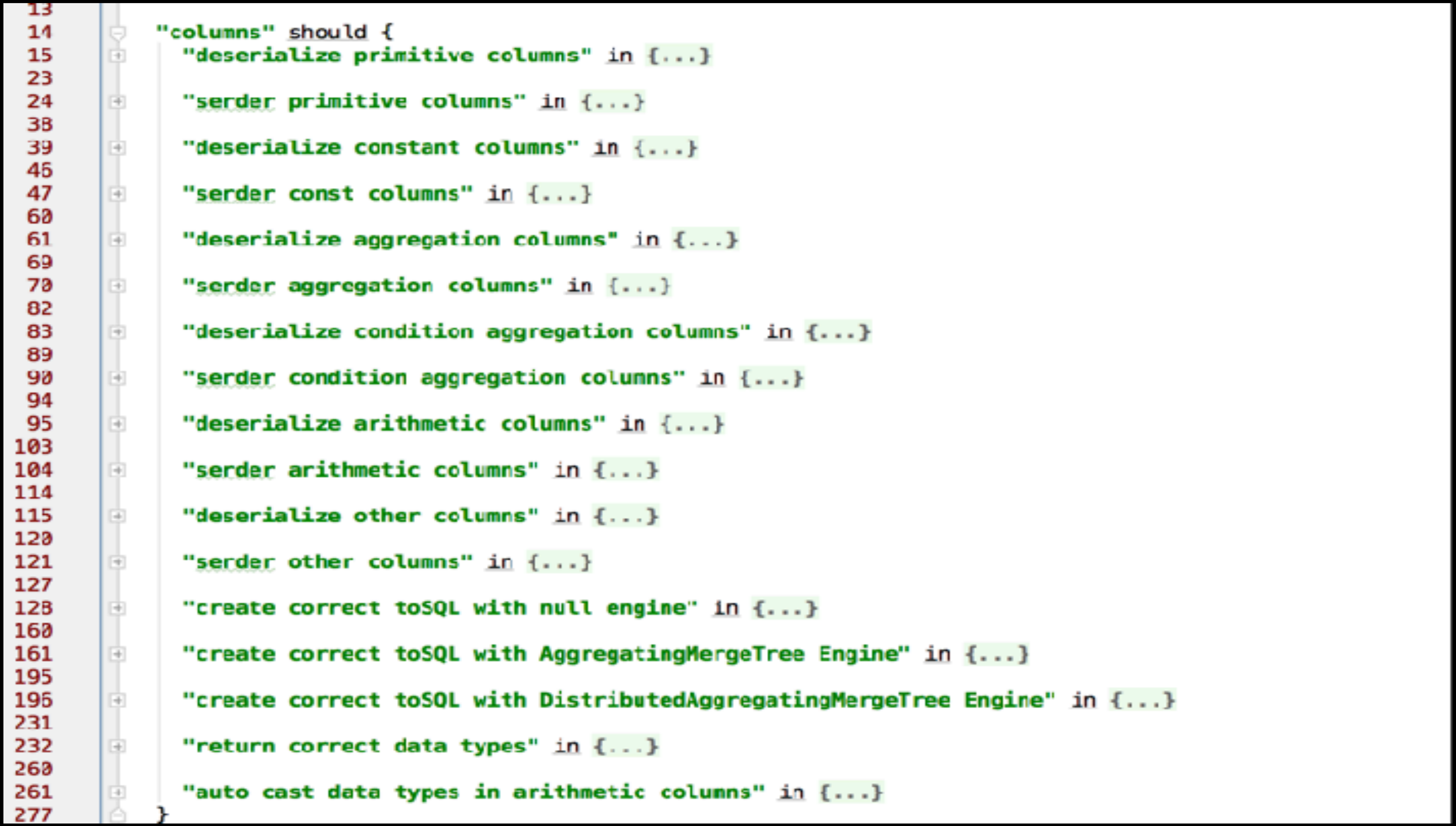
- 保证代码质量
- 保证重构的顺利进行
- 对代码的设计进行验证
- 保证项目可以持续性的维护下去

保证代码质量

- 怎样保证代码能够按照设想的方式正常运行?
- 怎样保证代码在异常情况下也能够正常的进行处理?

保证代码质量

- 我们可以编写测试对异常情况,正常流程进行测试



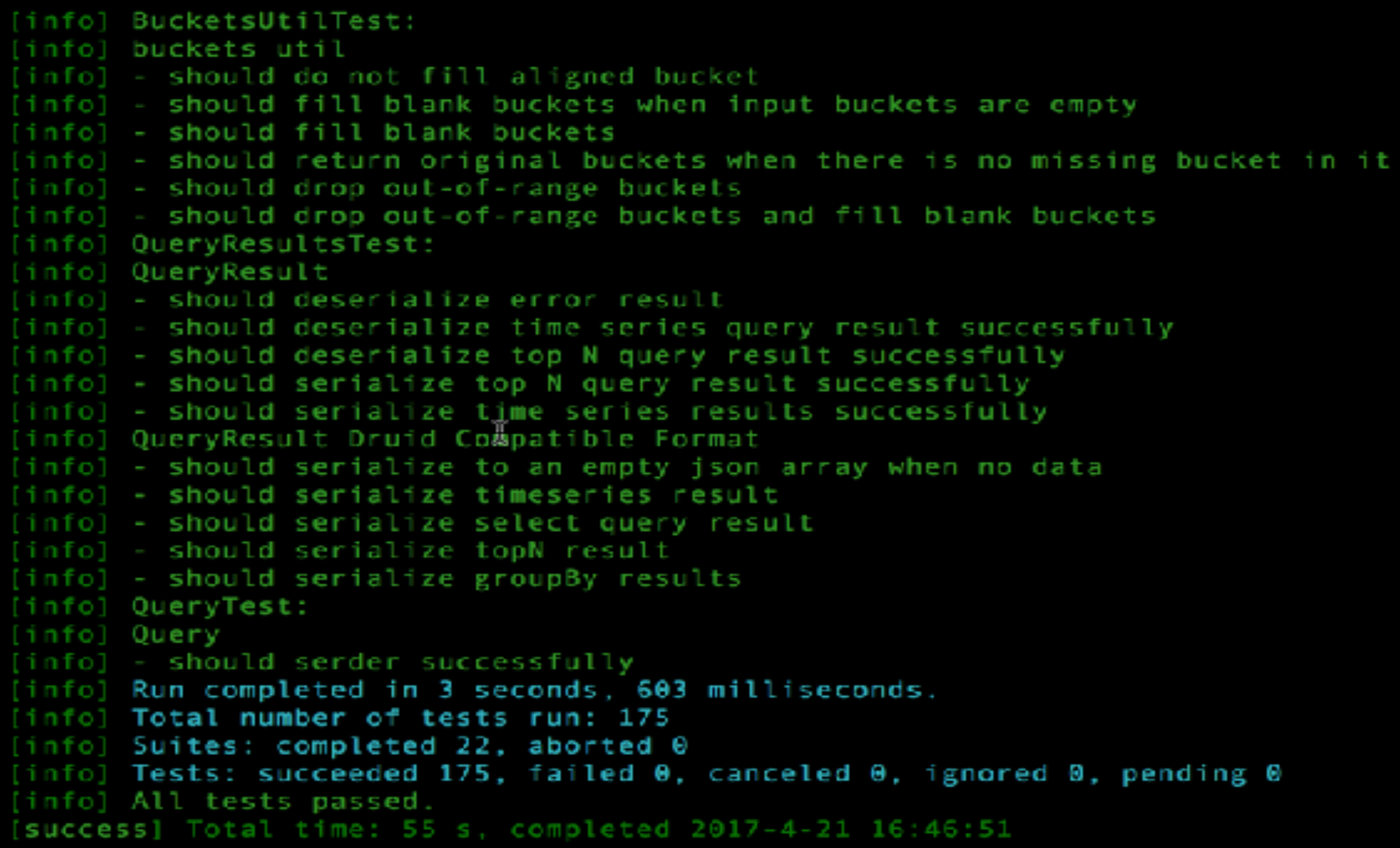
```
"columns" should {
  "deserialize primitive columns" in {...}

  "serder primitive columns" in {...}

  "deserialize constant columns" in {...}

  "serder const columns" in {...}

  "deserialize aggregation columns" in {...}

  "serder aggregation columns" in {...}

  "deserialize condition aggregation columns" in {...}

  "serder condition aggregation columns" in {...}

  "deserialize arithmetic columns" in {...}

  "serder arithmetic columns" in {...}

  "deserialize other columns" in {...}

  "serder other columns" in {...}

  "create correct toSQL with null engine" in {...}

  "create correct toSQL with AggregatingMergeTree Engine" in {...}

  "create correct toSQL with DistributedAggregatingMergeTree Engine" in {...}

  "return correct data types" in {...}

  "auto cast data types in arithmetic columns" in {...}
}
```

保证重构顺利

- 随着代码的不断增多,新加入的代码会不会搞挂现有功能?
- 重构的时候,会不会造成破坏?

保证重构的顺利

- 重新执行一下之前的测试代码



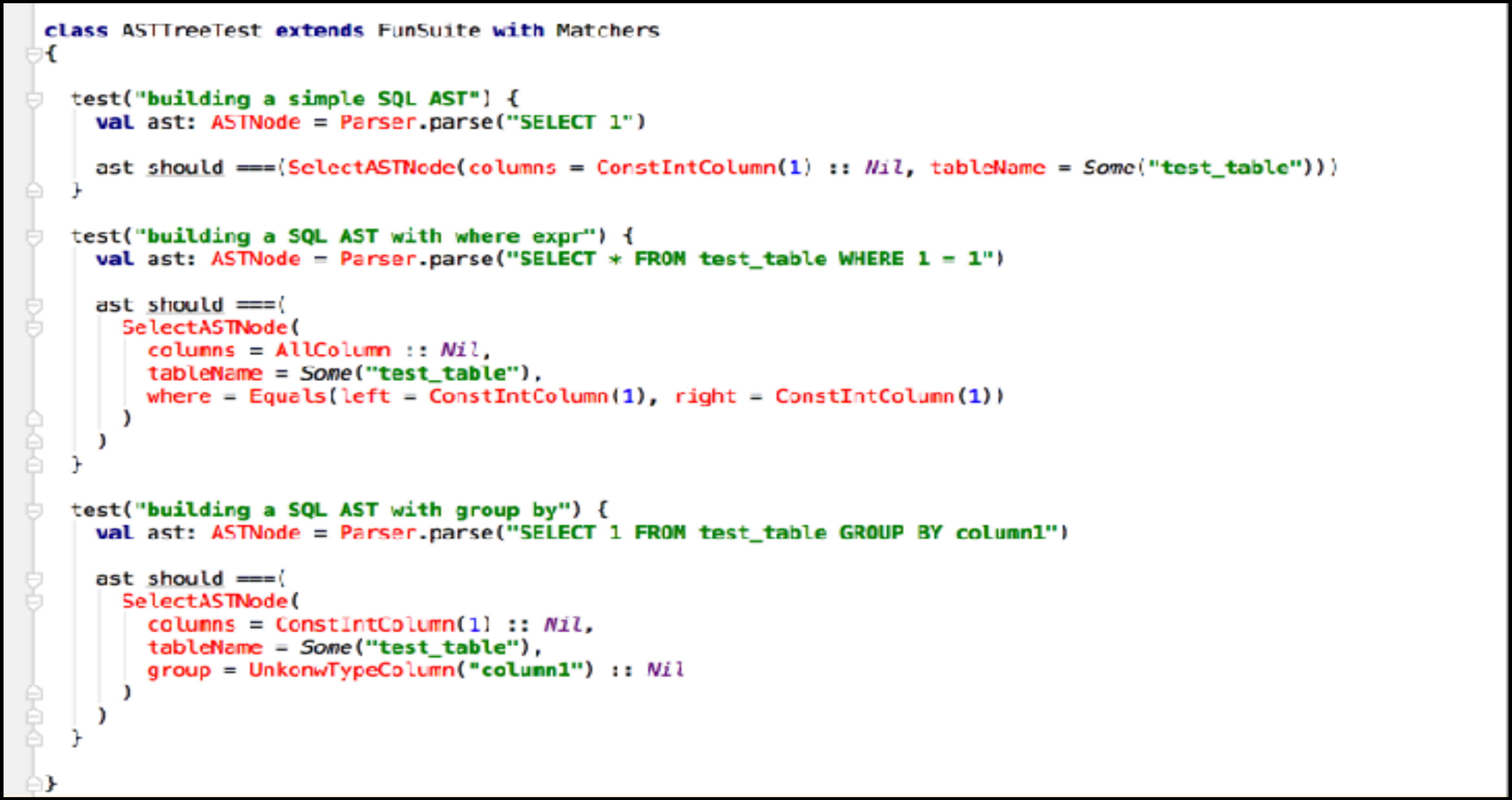
```
[info] BucketsUtilTest:
[info] buckets util
[info] - should do not fill aligned bucket
[info] - should fill blank buckets when input buckets are empty
[info] - should fill blank buckets
[info] - should return original buckets when there is no missing bucket in it
[info] - should drop out-of-range buckets
[info] - should drop out-of-range buckets and fill blank buckets
[info] QueryResultsTest:
[info] QueryResult
[info] - should deserialize error result
[info] - should deserialize time series query result successfully
[info] - should deserialize top N query result successfully
[info] - should serialize top N query result successfully
[info] - should serialize time series results successfully
[info] QueryResult Druid Compatible Format
[info] - should serialize to an empty json array when no data
[info] - should serialize timeseries result
[info] - should serialize select query result
[info] - should serialize topN result
[info] - should serialize groupBy results
[info] QueryTest:
[info] Query
[info] - should serder successfully
[info] Run completed in 3 seconds, 603 milliseconds.
[info] Total number of tests run: 175
[info] Suites: completed 22, aborted 0
[info] Tests: succeeded 175, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[success] Total time: 55 s, completed 2017-4-21 16:46:51
```

帮助模块设计

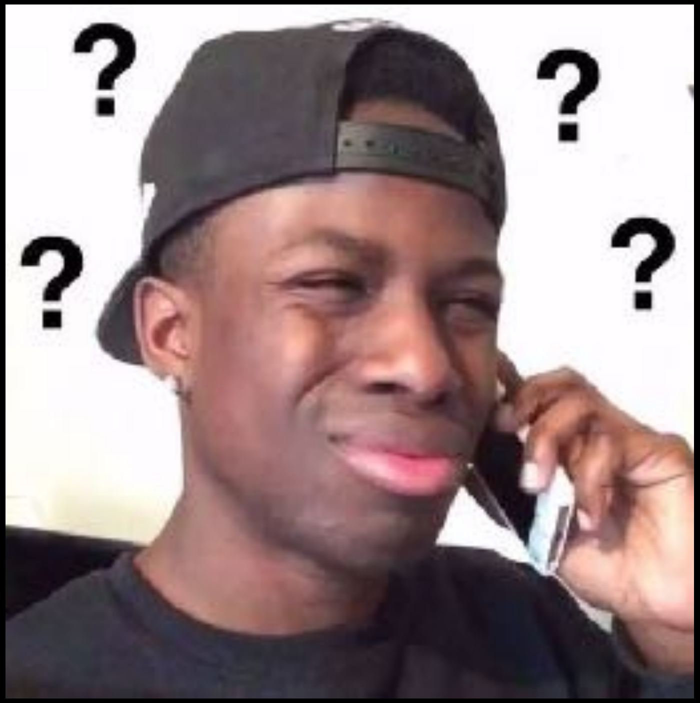
- 设计的模块是否足够简洁?
- 设计的模块是否具备可测性?
- 设计的模块是否可以帮助使用者方便实现功能?
- 实现结构是否足够简单,与其他模块耦合较低?

帮助模块设计

- 测试先行



```
class ASTTreeTest extends FunSuite with Matchers
{

  test("building a simple SQL AST") {
    val ast: ASTNode = Parser.parse("SELECT 1")

    ast should ===(SelectASTNode(columns = ConstIntColumn(1) :: Nil, tableName = Some("test_table")))
  }

  test("building a SQL AST with where expr") {
    val ast: ASTNode = Parser.parse("SELECT * FROM test_table WHERE 1 = 1")

    ast should ===(
      SelectASTNode(
        columns = AllColumn :: Nil,
        tableName = Some("test_table"),
        where = Equals(left = ConstIntColumn(1), right = ConstIntColumn(1))
      )
    )
  }

  test("building a SQL AST with group by") {
    val ast: ASTNode = Parser.parse("SELECT 1 FROM test_table GROUP BY column1")

    ast should ===(
      SelectASTNode(
        columns = ConstIntColumn(1) :: Nil,
        tableName = Some("test_table"),
        group = UnkonwTypeColumn("column1") :: Nil
      )
    )
  }

}
```



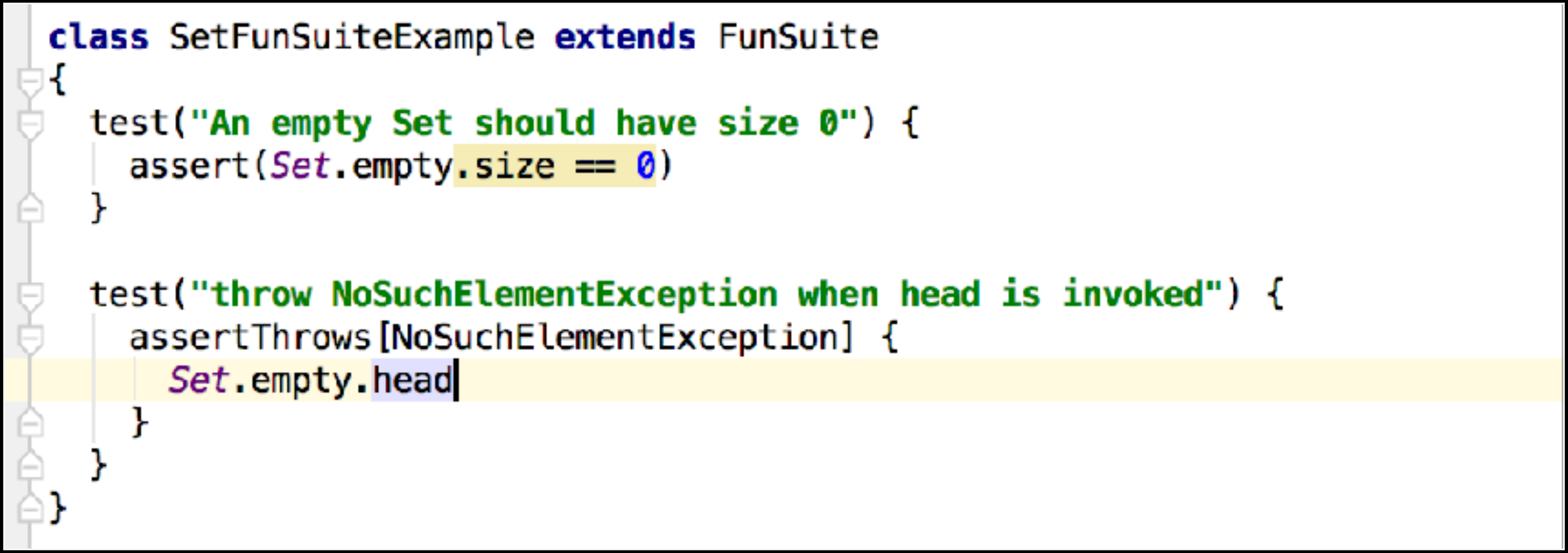
来点实际的吧!!

Scala Test Framework

ScalaTest

FunSuite

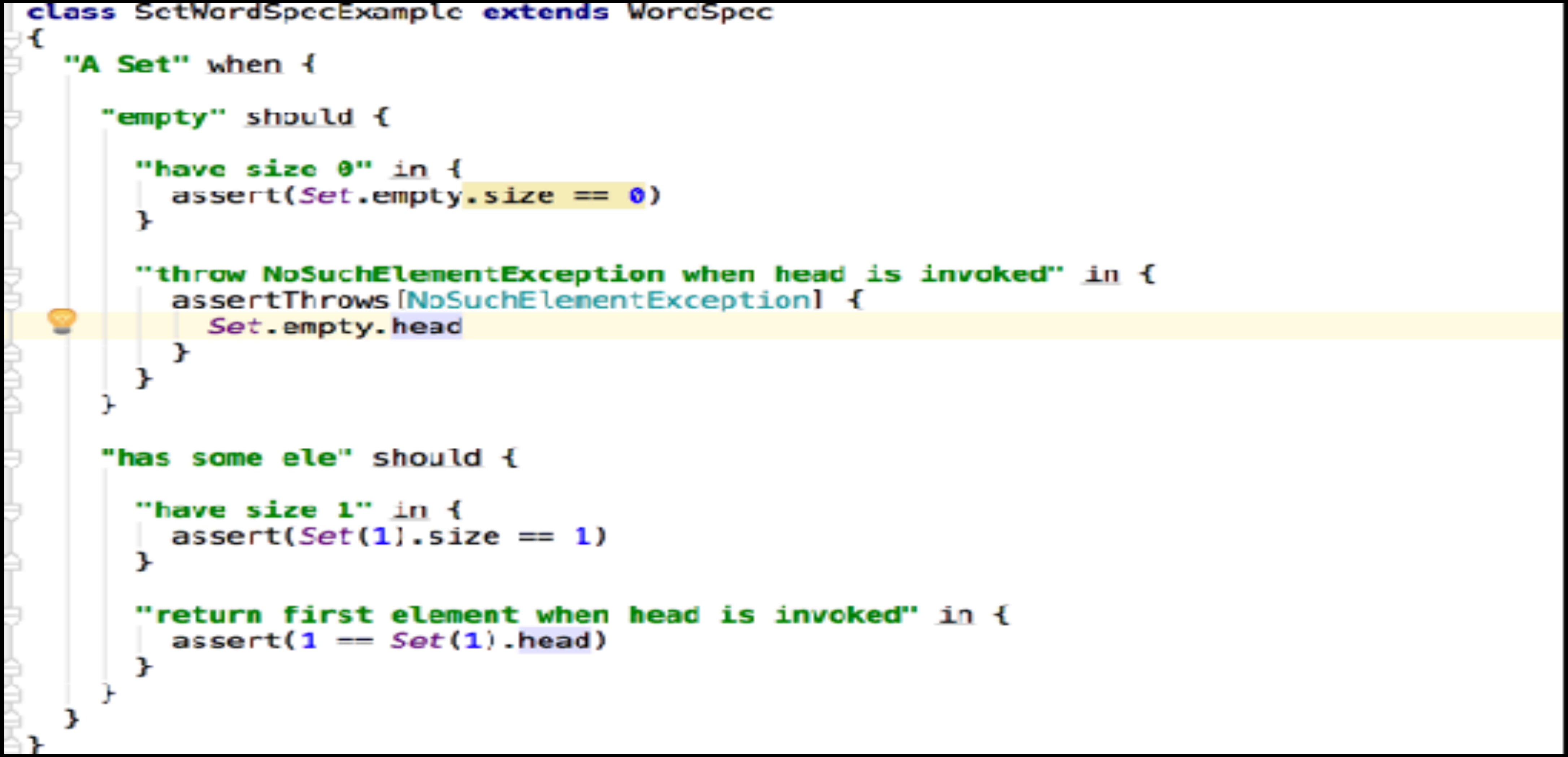
- 风格比较规范,类似一个方法给出一个测试场景,适合于编写单元测试



```
class SetFunSuiteExample extends FunSuite
{
  test("An empty Set should have size 0") {
    assert(Set.empty.size == 0)
  }

  test("throw NoSuchElementException when head is invoked") {
    assertThrows[NoSuchElementException] {
      Set.empty.head
    }
  }
}
```

WordSpec

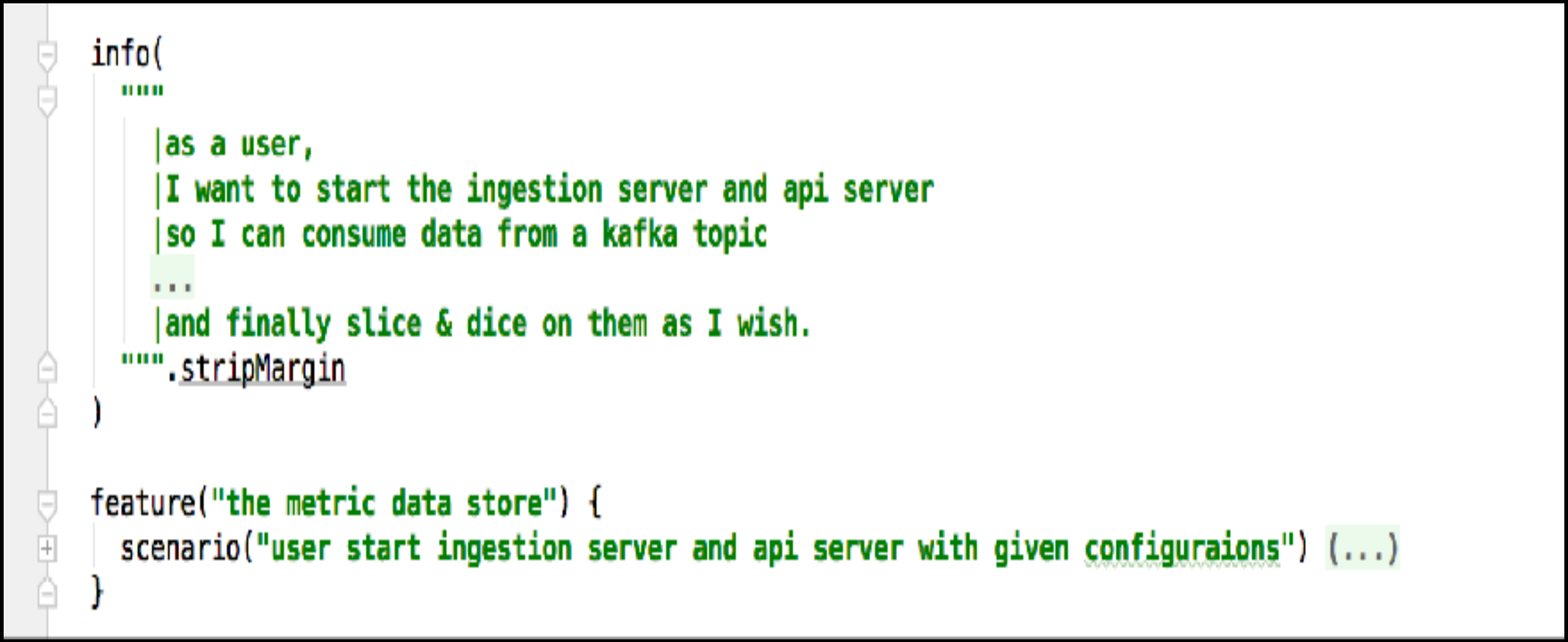
- 多测试用例组之间进行组合,可以实现针对一个复杂模块进行各条件性测试



```
class SetWordSpecExample extends WordSpec
{
  "A Set" when {

    "empty" should {

      "have size 0" in {
        assert(Set.empty.size == 0)
      }

      "throw NoSuchElementException when head is invoked" in {
        assertThrows[NoSuchElementException] {
          Set.empty.head
        }
      }
    }

    "has some ele" should {

      "have size 1" in {
        assert(Set(1).size == 1)
      }

      "return first element when head is invoked" in {
        assert(1 == Set(1).head)
      }
    }
  }
}
```

FeatureSpec

- 多测试用例组之间进行组合,可以实现针对一个复杂模块进行各条件性测试

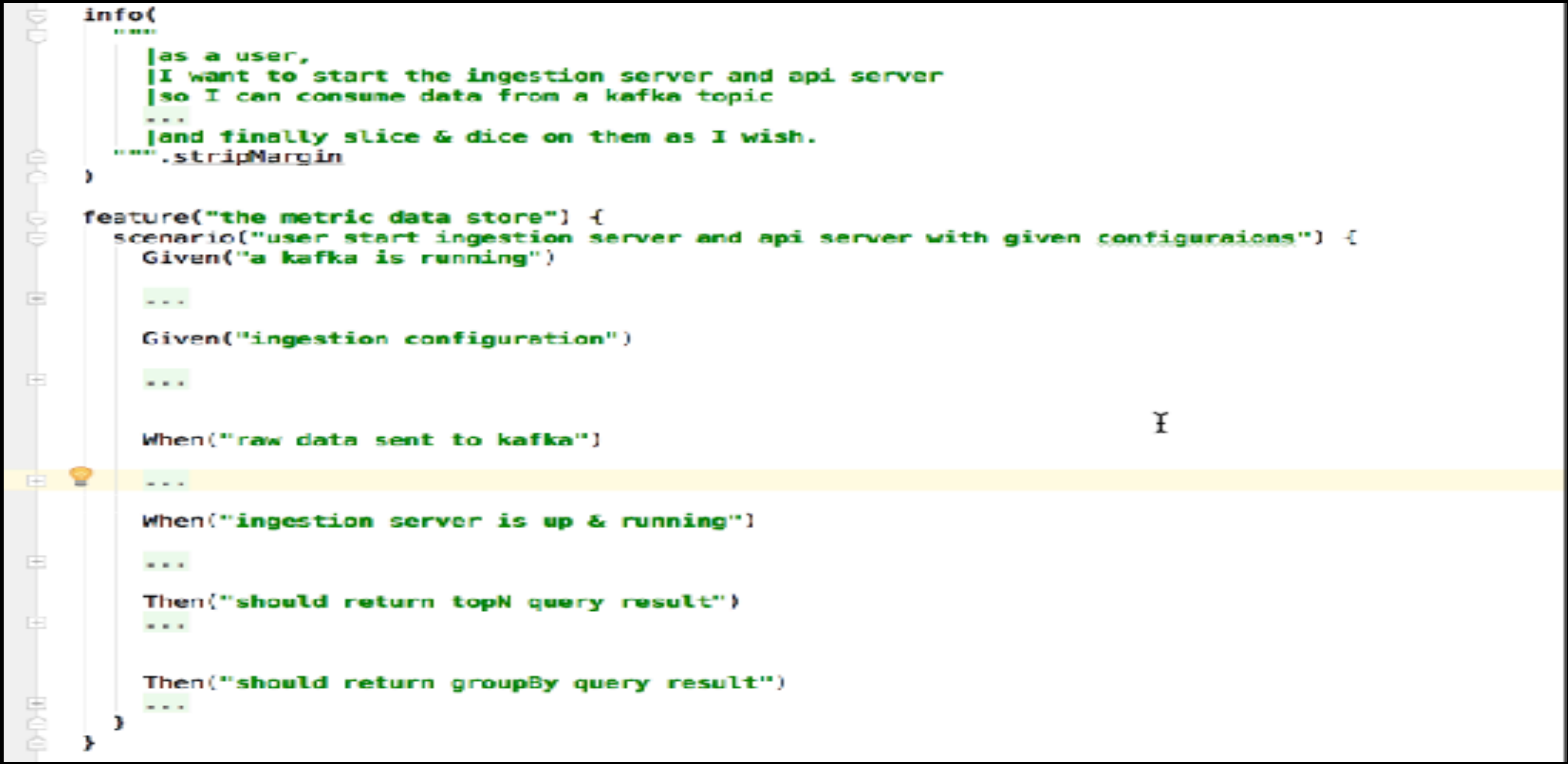


```
info(
  """
    |as a user,
    |I want to start the ingestion server and api server
    |so I can consume data from a kafka topic
   ...
    |and finally slice & dice on them as I wish.
  """.stripMargin
)

feature("the metric data store") {
  scenario("user start ingestion server and api server with given configuraions") {...}
}
```

FeatureSpec真有存在的必要么？

GivenWhenThen

- FeatureSpec + GivenWhenThen

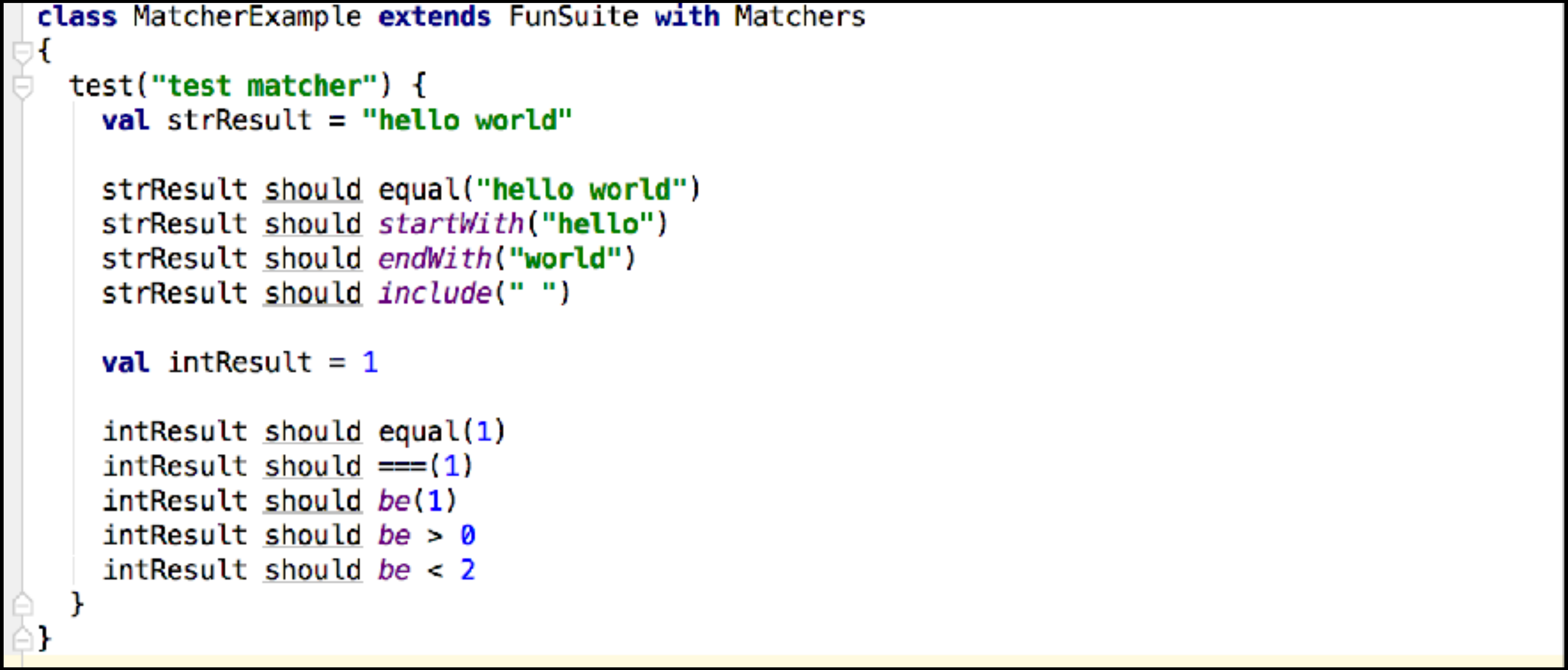


```
info(
  """
    |as a user,
    |I want to start the ingestion server and api server
    |so I can consume data from a kafka topic
    ...
    |and finally slice & dice on them as I wish.
  """.stripMargin
)

feature("the metric data store") {
  scenario("user start ingestion server and api server with given configuraions") {
    Given("a kafka is running")

    ...

    Given("ingestion configuration")

    ...


    When("raw data sent to kafka")

    ...

    When("ingestion server is up & running")

    ...

    Then("should return topN query result")
    ...


    Then("should return groupBy query result")
    ...
  }
}
```

仅仅这样么？可是…
我觉得我还有更复杂的验证需求…

Matchers

- Matechers提供非常多用于验证的断言种类



```scala
class MatcherExample extends FunSuite with Matchers
{
  test("test matcher") {
    val strResult = "hello world"

    strResult should equal("hello world")
    strResult should startWith("hello")
    strResult should endWith("world")
    strResult should include(" ")

    val intResult = 1

    intResult should equal(1)
    intResult should ===(1)
    intResult should be(1)
    intResult should be > 0
    intResult should be < 2
  }
}
```

“ScalaTest更多请参考”

http://www.scalatest.org/

Akka Test Framework

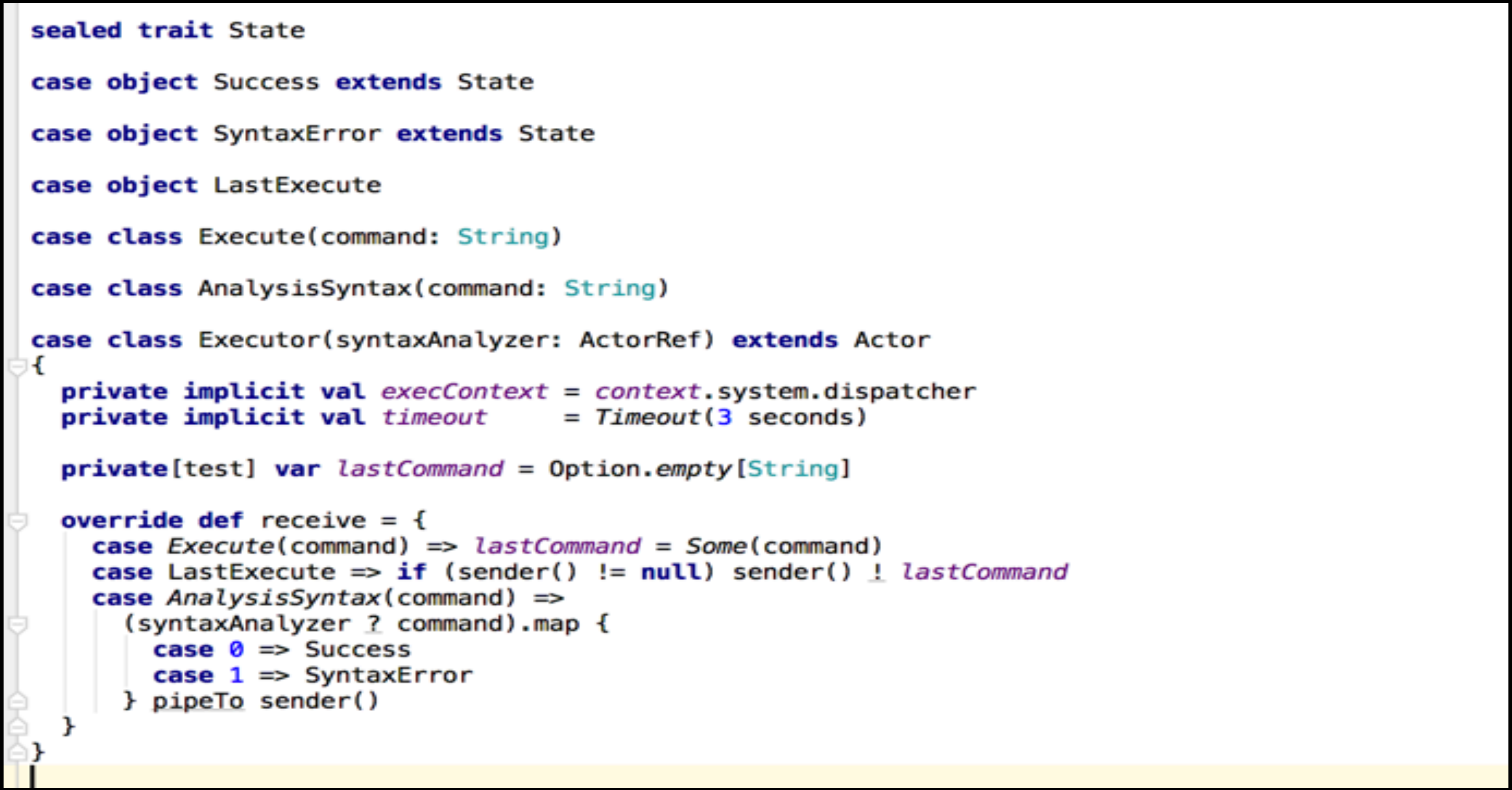
Akka-TestKit
&Akka-Stream-TestKit

“ *Akka*测试太困难？”

你还是太年轻~_~。。

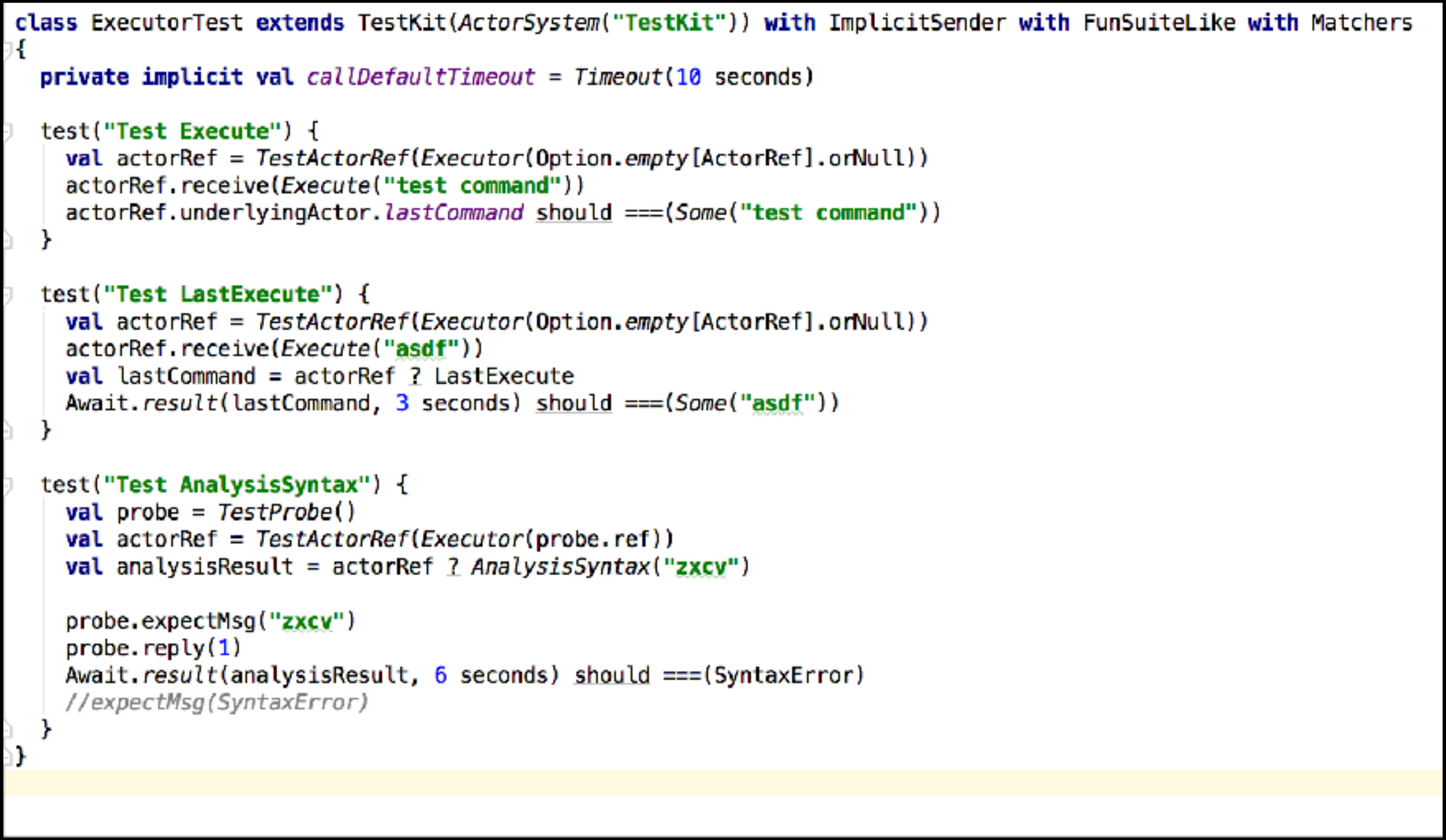
Actor

- 如何测试针对Message作出不同响应的Actor？



```scala
sealed trait State

case object Success extends State

case object SyntaxError extends State

case object LastExecute

case class Execute(command: String)

case class AnalysisSyntax(command: String)

case class Executor(syntaxAnalyzer: ActorRef) extends Actor
{
  private implicit val execContext = context.system.dispatcher
  private implicit val timeout     = Timeout(3 seconds)

  private[test] var lastCommand = Option.empty[String]

  override def receive = {
    case Execute(command) => lastCommand = Some(command)
    case LastExecute => if (sender() != null) sender() ! lastCommand
    case AnalysisSyntax(command) =>
      (syntaxAnalyzer ? command).map {
        case 0 => Success
        case 1 => SyntaxError
      } pipeTo sender()
  }
}
```

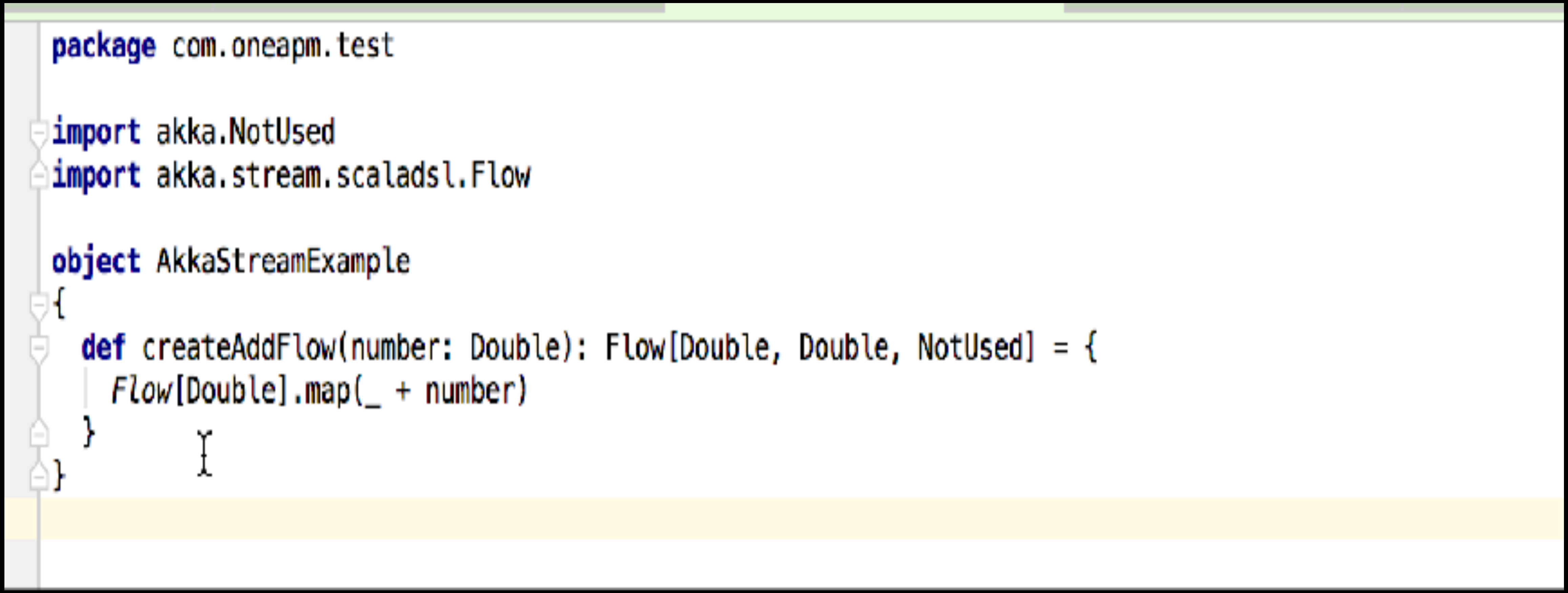
Actor



```
class ExecutorTest extends TestKit(ActorSystem("TestKit")) with ImplicitSender with FunSuiteLike with Matchers
{
  private implicit val callDefaultTimeout = Timeout(10 seconds)

  test("Test Execute") {
    val actorRef = TestActorRef(Executor(Option.empty[ActorRef].orNull))
    actorRef.receive(Execute("test command"))
    actorRef.underlyingActor.lastCommand should ===(Some("test command"))
  }

  test("Test LastExecute") {
    val actorRef = TestActorRef(Executor(Option.empty[ActorRef].orNull))
    actorRef.receive(Execute("asdf"))
    val lastCommand = actorRef ? LastExecute
    Await.result(lastCommand, 3 seconds) should ===(Some("asdf"))
  }

  test("Test AnalysisSyntax") {
    val probe = TestProbe()
    val actorRef = TestActorRef(Executor(probe.ref))
    val analysisResult = actorRef ? AnalysisSyntax("zxcv")

    probe.expectMsg("zxcv")
    probe.reply(1)
    Await.result(analysisResult, 6 seconds) should ===(SyntaxError)
    //expectMsg(SyntaxError)
  }
}
```

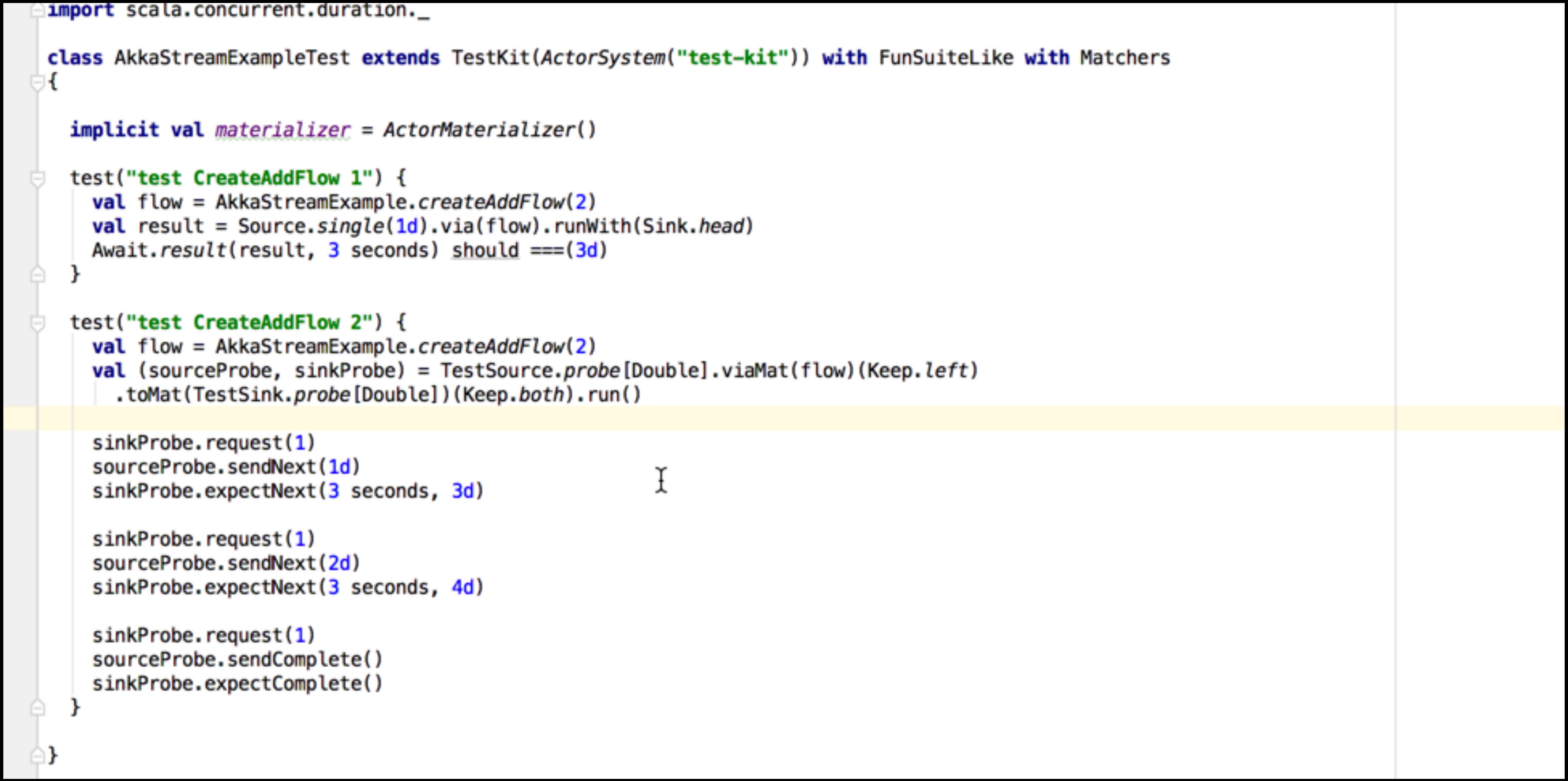
Stream

- 比方说,写出下列Flow的测试?



```scala
package com.oneapm.test

import akka.NotUsed
import akka.stream.scaladsl.Flow

object AkkaStreamExample
{
  def createAddFlow(number: Double): Flow[Double, Double, NotUsed] = {
    Flow[Double].map(_ + number)
  }
}
```

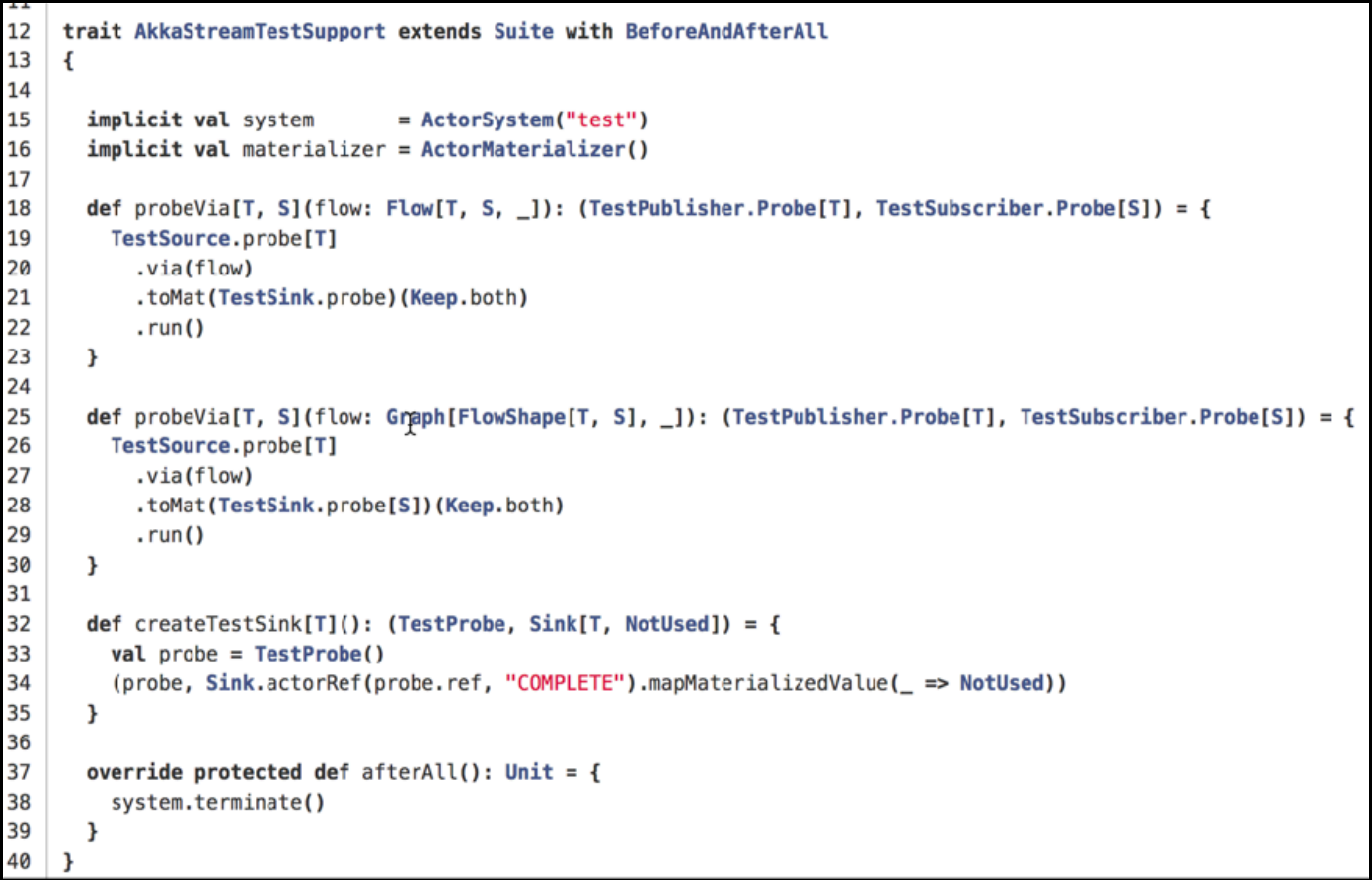
Stream



```scala
import scala.concurrent.duration._

class AkkaStreamExampleTest extends TestKit(ActorSystem("test-kit")) with FunSuiteLike with Matchers
{

  implicit val materializer = ActorMaterializer()

  test("test CreateAddFlow 1") {
    val flow = AkkaStreamExample.createAddFlow(2)
    val result = Source.single(1d).via(flow).runWith(Sink.head)
    Await.result(result, 3 seconds) should ===(3d)
  }

  test("test CreateAddFlow 2") {
    val flow = AkkaStreamExample.createAddFlow(2)
    val (sourceProbe, sinkProbe) = TestSource.probe[Double].viaMat(flow)(Keep.left)
      .toMat(TestSink.probe[Double])(Keep.both).run()

    sinkProbe.request(1)
    sourceProbe.sendNext(1d)
    sinkProbe.expectNext(3 seconds, 3d)

    sinkProbe.request(1)
    sourceProbe.sendNext(2d)
    sinkProbe.expectNext(3 seconds, 4d)

    sinkProbe.request(1)
    sourceProbe.sendComplete()
    sinkProbe.expectComplete()
  }

}
```

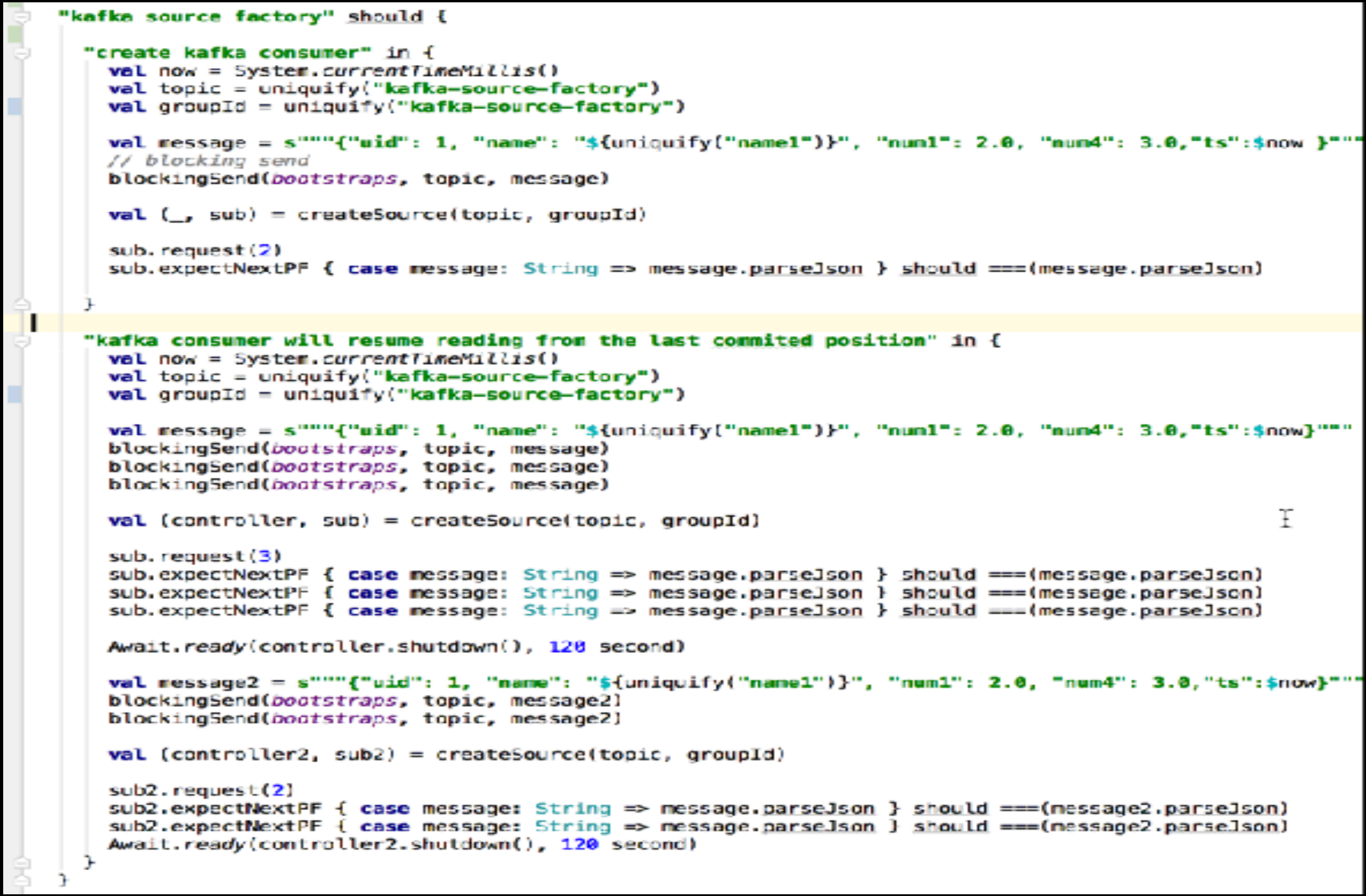
我们的使用

StreamSupport



```scala
trait AkkaStreamTestSupport extends Suite with BeforeAndAfterAll
{

  implicit val system       = ActorSystem("test")
  implicit val materializer = ActorMaterializer()

  def probeVia[T, S](flow: Flow[T, S, _]): (TestPublisher.Probe[T], TestSubscriber.Probe[S]) = {
    TestSource.probe[T]
      .via(flow)
      .toMat(TestSink.probe)(Keep.both)
      .run()
  }

  def probeVia[T, S](flow: Graph[FlowShape[T, S], _]): (TestPublisher.Probe[T], TestSubscriber.Probe[S]) = {
    TestSource.probe[T]
      .via(flow)
      .toMat(TestSink.probe[S])(Keep.both)
      .run()
  }

  def createTestSink[T](): (TestProbe, Sink[T, NotUsed]) = {
    val probe = TestProbe()
    (probe, Sink.actorRef(probe.ref, "COMPLETE").mapMaterializedValue(_ => NotUsed))
  }

  override protected def afterAll(): Unit = {
    system.terminate()
  }
}
```

Kafka Source



```
"kafka source factory" should {

  "create kafka consumer" in {
    val now = System.currentTimeMillis()
    val topic = uniquify("kafka-source-factory")
    val groupId = uniquify("kafka-source-factory")

    val message = s"""{"uid": 1, "name": "${uniquify("name1")}", "num1": 2.0, "num4": 3.0,"ts":$now }""
    // blocking send
    blockingSend(bootstraps, topic, message)

    val (_, sub) = createSource(topic, groupId)

    sub.request(2)
    sub.expectNextPF { case message: String => message.parseJson } should ===(message.parseJson)

  }

  "kafka consumer will resume reading from the last commited position" in {
    val now = System.currentTimeMillis()
    val topic = uniquify("kafka-source-factory")
    val groupId = uniquify("kafka-source-factory")

    val message = s"""{"uid": 1, "name": "${uniquify("name1")}", "num1": 2.0, "num4": 3.0,"ts":$now}"""
    blockingSend(bootstraps, topic, message)
    blockingSend(bootstraps, topic, message)
    blockingSend(bootstraps, topic, message)

    val (controller, sub) = createSource(topic, groupId)

    sub.request(3)
    sub.expectNextPF { case message: String => message.parseJson } should ===(message.parseJson)
    sub.expectNextPF { case message: String => message.parseJson } should ===(message.parseJson)
    sub.expectNextPF { case message: String => message.parseJson } should ===(message.parseJson)

    Await.ready(controller.shutdown(), 120 second)

    val message2 = s"""{"uid": 1, "name": "${uniquify("name1")}", "num1": 2.0, "num4": 3.0,"ts":$now}""
    blockingSend(bootstraps, topic, message2)
    blockingSend(bootstraps, topic, message2)

    val (controller2, sub2) = createSource(topic, groupId)

    sub2.request(2)
    sub2.expectNextPF { case message: String => message.parseJson } should ===(message2.parseJson)
    sub2.expectNextPF { case message: String => message.parseJson } should ===(message2.parseJson)
    Await.ready(controller2.shutdown(), 120 second)
  }
}
```

“AkkaTest更多请参考”

http://akka.io/docs/

演示

谢谢