Performing Expressive Testing Using Matchers

Harit Himanshu
FOUNDER, [BONSAIILABS.COM](http://bonsaiilabs.com)

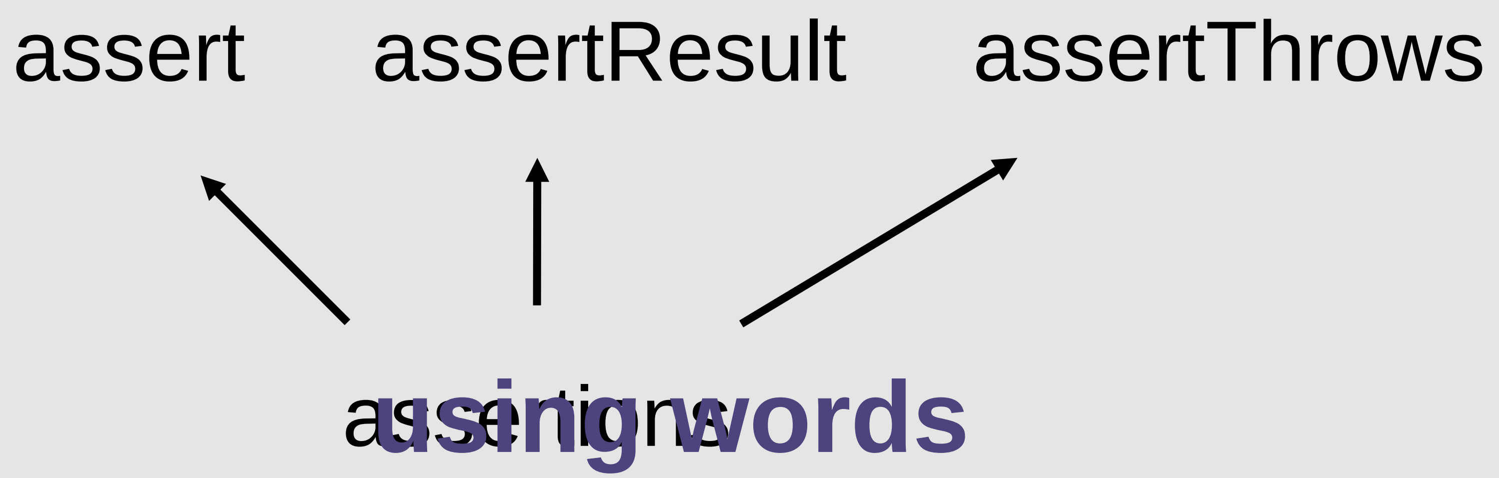
@harittweets   www.bonsaiilabs.com

Agenda

- Understanding *Matchers*
- Using *Matchers* to Test Equality
- Using *Matchers* to Test Strings
- Using *Matchers* to Test Elements Ordering
- Creating a *Base Test* for Unit Tests
- Using *Matchers* to Test for Length & Size
- Using *Matchers* to Test Container Elements
- Using *Matchers* to Test Emptiness Property
- Using *Matchers* to Test Exceptions
- Using *Matchers* to Test Logical Operations
- Using *Matchers* to Test Negative Statement Structure
- Using *Matchers* to Test Object Identity
- Using *Matchers* to Test Pattern Matching

Understanding Matchers

using words



Matchers Examples

currency1 **shouldEqual** currency2

wallet **should not contain** (hundredInr)

customerIds **should have size** 2

Domain-Specific Language (DSL)

```
import org.scalatest._
import Matchers._

class EmailSpec extends FlatSpec {
 // matchers are available now
}
```

Importing all Matchers

```scala
import org.scalatest._

class EmailSpec extends FlatSpec with Matchers {
 // matchers are available now
}
```

Mixing The **Matchers** Trait

Using Matchers to Test Equality

$$\begin{tikzcd}
\text{left} \; \textbf{should equal} \; \text{(right)} \\
\text{org.scalactic.Equality[L]} \arrow[u, dashed, "\text{typeof}"']
\end{tikzcd}$$

org.scalactic.Equality[String]

“Hello” **should equal** (“Hello”)

org.scalactic.Equality[Int]

100 **should equal** (100)

Type (T)	Equality[T]	Example

String	Equality[String]	`scala> "Hello" == "Hello"`
		`res1: Boolean = true`

Int	Equality[Int]	`scala> 100 == 100`
		`res2: Boolean = true`

Boolean	Equality[Boolean]	`scala> false == false`
		`res3: Boolean = true`

Double	Equality[Double]	`scala> 10.01 == 10.01`
		`res5: Boolean = true`

Char	Equality[Char]	`scala> 'A' == 'A'`
		`res0: Boolean = true`

Case Class	Equality[Apple]	`scala> case class Fruit(color: String, taste: String)`
		`defined class Fruit`
		
		`scala> val apple = Fruit("red", "sweet")`
		`apple: Fruit = Fruit(red,sweet)`
		
		`scala> apple == apple`
		`res4: Boolean = true`

Arrays as Exception

```
scala> Array(100, 200) == Array(100, 200)
res6: Boolean = false
```

```
Array(100, 200) should equal (Array(100, 200)) // success
```

Equality Test Syntax

```
"hello" should equal ("hello")

"hello" should === ("hello")

"hello" should be ("hello")
```

Customize Equality

type check at compile time

```
"hello" shouldEqual (“hello")

"hello" shouldBe (“hello")
```

Cannot
Customize Equality

Equality[T]

Using Matchers to Test Strings

MatcherWords.scala

```
"hello" should startWith ("h")

 "hello" should endWith ("o")

 "hello" should include ("e")
```

Using Matchers to Test Container Elements

Containing Syntax

```
List(1, 2) should contain (1)
```

```
org.scalatest.enablers.Containing[L]
```

$$\begin{tikzcd}
\texttt{Containing[L]} \arrow[d] \\
\texttt{Containing[Set[Int]]} \arrow[d] \\
\texttt{Equality[Int]}
\end{tikzcd}$$

```
GenTraversable[E]
java.util.Collection[E]
java.util.Map[K, V]
String
Array[E]
Option[E]
```

Using Matchers to Test Emptiness Property

Emptiness Syntax

```
List() should be (empty)
```

```
T => Emptiness[T]
```

```
GenTraversable[E]
java.util.Collection[E]
java.util.Map[K, V]
String
Array[E]
Option[E]
```

Summary

- Understanding *Matchers*
- Using *Matchers* to Test Equality
- Using *Matchers* to Test Strings
- Using *Matchers* to Test Elements Ordering
- Creating a *Base Test* for Unit Tests
- Using *Matchers* to Test for Length & Size
- Using *Matchers* to Test Container Elements
- Using *Matchers* to Test Emptiness Property
- Using *Matchers* to Test Exceptions
- Using *Matchers* to Test Logical Operations
- Using *Matchers* to Test Negative Statement Structure
- Using *Matchers* to Test Object Identity
- Using *Matchers* to Test Pattern Matching