5.– 8. September 2011
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Ran an die Spec

Ausführbare Software-Spezifikationen mit specs²

Andreas Flierl
imbus AG

„Any sufficiently primitive magic is indistinguishable from technology.“

variation of Arthur C. Clarke‘s third law

… wth?

- Java-Entwickler (J2EE + Swing)
- agil
- clean code
- Scala-Enthusiast
- specs2-Fan

„specs2 is a library for writing executable software specifications.“

Eric Torreborre (author of specs2)

behaviour-driven

fly-by

setup & running

structure

matchers

roundup

behaviour-driven

fly-by

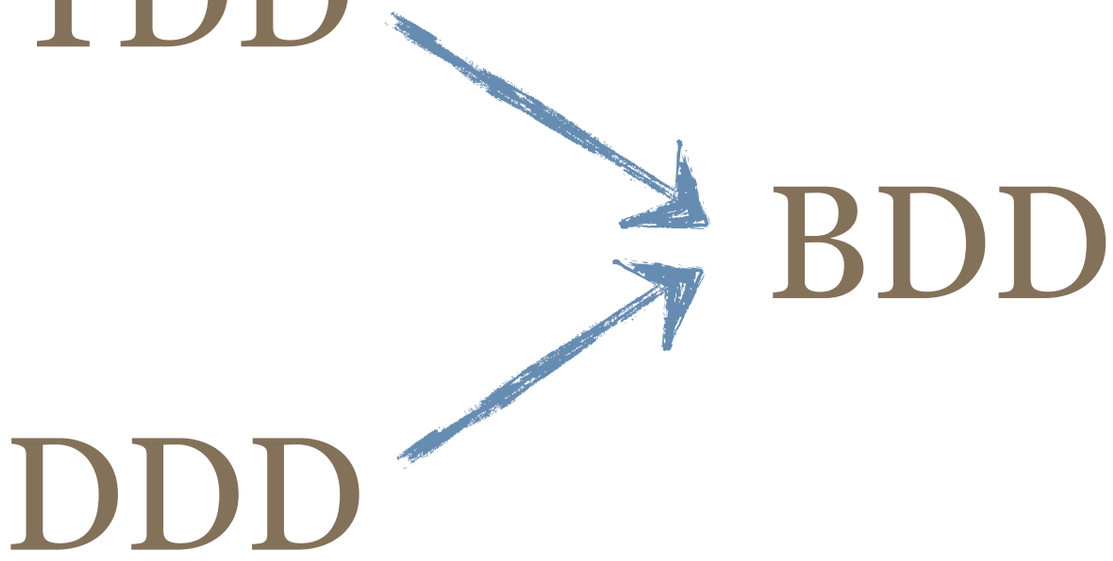
setup & running

structure

matchers

roundup

Behaviour-Driven Development



Behaviour-Driven Development

TDD

DDD

→ BDD

- die richtigen Worte finden
- agile

Ausführbare Software-Spezifikation

- informell
- verhaltensorientiert
- Code wie Literatur
- „literate“

Ausführbare Software-Spezifikation

- informell
- verhaltensorientiert
- Code wie Literatur
- „literate“

- D. E. Knuth (1984)
- R. C. Martin (2008)

… im Sinne von specs²

- selbstbeschreibende Tests
- gewünschtes Verhalten
- Zweck
- *verständliche* Ausgabe

… im Sinne von specs²

- selbstbeschreibende Tests
- gewünschtes Verhalten
- Zweck
- *verständliche* Ausgabe

für alle Beteiligten

behaviour-driven

fly-by

setup & running

structure

matchers

roundup

Zwei verschiedene Bauweisen

- unit spec
- acceptance spec

unit spec

```
import org.specs2.mutable._

object FluxCapacitorSpec extends Specification {
  "The flux capacitor" should {
    "be carried by a DeLorean" in {
      FluxCapacitor.carrier must be equalTo "DeLorean DMC-12"
    }

    "require approx. 1.21 gigawatts of electrical power" in {
      FluxCapacitor.requiredPower must be closeTo(1.21e9 +/- 1e6)
    }

    "not yet be able to use the MrFusion power converter" in {
      FluxCapacitor.use(MrFusion) must throwA[SciFiException]
    }
  }
}
```

unit spec

- Beschreibung und Code verwoben
- „should“ und „in“
- leichte Einschränkungen
- Veränderbarkeit (mutability)
- Seiteneffekte

acceptance spec

```
import org.specs2._

object FluxCapacitorSpec extends Specification { def is =
  "The flux capacitor should"                                          ^
    "be carried by a DeLorean"                                       ! e1^
    "require approx. 1.21 gigawatts of electrical power"             ! e2^
    "not yet be able to use the MrFusion power converter"            ! e3

  def e1 = FluxCapacitor.carrier must be equalTo "DeLorean DMC-12"
  def e2 = FluxCapacitor.requiredPower must be closeTo(1.21e9 +/- 1e6)
  def e3 = FluxCapacitor.use(MrFusion) must throwA[SciFiException]
}
```

acceptance spec

- Beschreibung verweist auf Code
- ^ und !
- voller Funktionsumfang
- referenzielle Transparenz
- Unveränderbarkeit (immutability)

Inline-Beispiele

```
import org.specs2._

object FluxCapacitorSpecInline extends Specification { def is =
  "The flux capacitor should" ^
    "be carried by a DeLorean" ! {
      FluxCapacitor.carrier must be equalTo "DeLorean DMC-12"
    } ^
    "require approx. 1.21 gigawatts of electrical power" ! {
      FluxCapacitor.requiredPower must be closeTo(1.21e9 +/- 1e6)
    } ^
    "not yet be able to use the MrFusion power converter" ! {
      FluxCapacitor.use(MrFusion) must throwA[SciFiException]
    }
}
```

Inline-Beispiele

```
import org.specs2._

object FluxCapacitorSpecInline extends Specification { def is =
  "The flux capacitor should" ^
    "be carried by a DeLorean" ! {
      FluxCapacitor.carrier must be equalTo "DeLorean DMC-12"
    } ^
    "require approx. 1.21 gigawatts of electrical power" ! {
      FluxCapacitor.requiredPower must be closeTo(1.21e9 +/- 1e6)
    } ^
    "not yet be able to use the MrFusion power converter" ! {
      FluxCapacitor.use(MrFusion) must throwA[SciFiException]
    }
}
```

Inline-Beispiele

```
import org.specs2._

object FluxCapacitorSpecInline extends Specification { def is =
  "The flux capacitor should" ^
    "be carried by a DeLorean" ! {
      FluxCapacitor.carrier must be equalTo "DeLorean DMC-12"
    } ^
    //
    "require approx. 1.21 gigawatts of electrical power" ! {
      FluxCapacitor.requiredPower must be closeTo(1.21e9 +/- 1e6)
    } ^
    //
    "not yet be able to use the MrFusion power converter" ! {
      FluxCapacitor.use(MrFusion) must throwA[SciFiException]
    }
}
```

Ergebnis (Konsole)

```
scala> specs2.run(FluxCapacitorSpec)
FluxCapacitorSpec
The flux capacitor should
+ be carried by a DeLorean
x require approx. 1.21 gigawatts of electrical power
    '1.22E9' is not close to 1.21E9 +/- 1000000.0 (FluxCapacitorSpec.scala:13)
+ not yet be able to use the MrFusion power converter


Total for specification FluxCapacitorSpec
Finished in 185 ms
3 examples, 1 failure (+1), 0 error
```

Ergebnis (HTML)

FluxCapacitorSpec *(issues only)*

The flux capacitor should
- be carried by a DeLorean
- require approx. 1.21 gigawatts of electrical power
  '1.22E9' is not close to 1.21E9 +/- 1000000.0 (FluxCapacitorSpec.scala:9)
- not yet be able to use the MrFusion power converter

Total for specification FluxCapacitorSpec	
Finished in	168 ms
Results	3 examples, 1 failure, 0 error

```
spec.content |> select |> sequence |> execute |> store |> export(spec)
```

behaviour-driven

fly-by

setup & running

structure

matchers

roundup

Tool-Support

- sbt (0.7.x oder > 0.9)
- JUnit
- REPL / Kommandozeile
- IntelliJ IDEA
- NotifierRunner

sbt (> 0.9)

```
libraryDependencies +=
  "org.specs2" %% "specs2" % "1.6" % "test"
```

sbt (> 0.9)

```
libraryDependencies ++= Seq(
  "org.specs2" %% "specs2" % "1.6" % "test",
  //... optional dependencies
)

testOptions := Seq(
  Tests.Filter(_ == "FluxCapacitorSpec"),
  Tests.Argument("html", "console", "junitxml"))

testOptions <+= crossTarget map { ct =>
  Tests.Setup { () =>
    System.setProperty("specs2.outDir",
                       new File(ct, "specs2").getAbsolutePath)
  }
}
```

sbt (> 0.9)

```
> test
...

> test-only FluxCapacitorSpec -- html console xonly
...
```

JUnit

```
import org.specs2._

class FluxCapacitorSpec extends SpecificationWithJUnit { def is =
  //...
}
```

oder

```
import org.specs2._
import org.specs2.runner.JUnitRunner
import org.junit.runner.RunWith

@RunWith(classOf[JUnitRunner])
class FluxCapacitorSpec extends Specification { def is =
  //...
}
```

JUnit

```
import org.specs2._

class FluxCapacitorSpec extends SpecificationWithJUnit { def is =
  //...
}
```

oder

```
import org.specs2._
import org.specs2.runner.JUnitRunner
import org.junit.runner.RunWith

@RunWith(classOf[JUnitRunner])
class FluxCapacitorSpec extends Specification { def is =
  //...
}
```

Scala – specs2tour/src/test/scala/eu/flierl/specs2tour/acceptance/FluxCapacitorSpec.scala – Eclipse – /Users/i0n/Documents/workspace

FluxCapacitorSpec.scala

```
1  package eu.flierl.specs2tour
2  package acceptance
3
4  import org.specs2._
5
6  class FluxCapacitorSpec extends SpecificationWithJUnit { def is =
7    "The flux capacitor should"                                  ^
8      "be carried by a DeLorean"                              ! e1^
9      "require approx. 1.21 gigawatts of electrical power"    ! e2^
10     "not yet be able to use the MrFusion power converter"   ! e3
11
12   def e1 = FluxCapacitor.carrier must be equalTo "DeLorean DMC-12"
13   def e2 = FluxCapacitor.requiredPower must be closeTo(1.21e9 +/- 1e6)
14   def e3 = FluxCapacitor.use(MrFusion) must throwA[SciFiException]
15 }
16
```

Outline

- eu.flierl.specs2tour.acceptance
- import declarations
- FluxCapacitorSpec
  - is: org.specs2.specification.Fragments
  - e1: org.specs2.matcher.MatchResult[java.lang.String]
  - e2: org.specs2.matcher.MatchResult[Double]
  - e3: org.specs2.matcher.MatchResult[java.lang.String]

Problems | Tasks | Console | JUnit

Finished after 0,117 seconds

Runs: 3/3 Errors: 0 Failures: 1

- FluxCapacitorSpec [Runner: JUnit 4] (0,049 s)
  - The flux capacitor should (0,049 s)
    - be carried by a DeLorean (0,002 s)
    - require approx. 1.21 gigawatts of electrical power (0,039 s)
    - not yet be able to use the MrFusion power converter (0,000 s)

Failure Trace

```
java.lang.Exception: '1.22E9' is not close to 1.21E9 +/- 1000000.0
at eu.flierl.specs2tour.acceptance.FluxCapacitorSpec.e2(FluxCapacitorSpec.scala:1
at eu.flierl.specs2tour.acceptance.FluxCapacitorSpec$$anonfun$is$1$$anonfun$a
at eu.flierl.specs2tour.acceptance.FluxCapacitorSpec$$anonfun$is$1$$anonfun$a
```

Writable | Smart Insert | 13 : 3 | 289M of 518M

REPL / Kommandozeile

```
> scala -cp ... specs2.run FluxCapacitorSpec [arg1 arg2 ...]
...

> scala -cp ... specs2.html FluxCapacitorSpec [arg1 arg2 ...]
...

> scala -cp ... specs2.junitxml FluxCapacitorSpec [arg1 arg2 ...]
...

> scala -cp ... org.specs2.files [arg1 arg2 ...]
...
```

Weitere

- IntelliJ IDEA
  - Spezifikation oder Beispiel auswählen
  - Ctrl+Shift+F10
  - Test options
- NotifierRunner
  - Trait „Notifier“ für Events

behaviour-driven

fly-by

setup & running

structure

matchers

roundup

Fragmente

- Spezifikation 

Fragment
- Optionen
- Freitext
- Beispiel (example) = Text + Code
- Aktion/Schritt (action/step)
- Etiketten (tags)
- SpecStart/SpecEnd

acceptance spec

```
import org.specs2._

object FluxCapacitorSpec extends Specification { def is =
  "The flux capacitor should"                                          ^
    "be carried by a DeLorean"                                       ! e1^
    "require approx. 1.21 gigawatts of electrical power"             ! e2^
    "not yet be able to use the MrFusion power converter"            ! e3

  def e1 = FluxCapacitor.carrier must be equalTo "DeLorean DMC-12"
  def e2 = FluxCapacitor.requiredPower must be closeTo(1.21e9 +/- 1e6)
  def e3 = FluxCapacitor.use(MrFusion) must throwA[SciFiException]
}
```

acceptance spec

```
import org.specs2._

object FluxCapacitorSpec extends Specification { def is =
  "The flux capacitor should"                                          ^
    "be carried by a DeLorean"                                       ! e1^
    "require approx. 1.21 gigawatts of electrical power"             ! e2^
    "not yet be able to use the MrFusion power converter"            ! e3

  def e1 = FluxCapacitor.carrier must be equalTo "DeLorean DMC-12"
  def e2 = FluxCapacitor.requiredPower must be closeTo(1.21e9 +/- 1e6)
  def e3 = FluxCapacitor.use(MrFusion) must throwA[SciFiException]
}
```

acceptance spec

```
import org.specs2._

object FluxCapacitorSpec extends Specification { def is =
  "The flux capacitor should"                                          ^
    "be carried by a DeLorean"                                         ! e1^
    "require approx. 1.21 gigawatts of electrical power"               ! e2^
    "not yet be able to use the MrFusion power converter"              ! e3

  def e1 = FluxCapacitor.carrier must be equalTo "DeLorean DMC-12"
  def e2 = FluxCapacitor.requiredPower must be closeTo(1.21e9 +/- 1e6)
  def e3 = FluxCapacitor.use(MrFusion) must throwA[SciFiException]
}
```

Ergebnisse

- Matcher-Ergebnis (MatchResult)
- Standard-Ergebnis
  - success / done
  - failure
  - anError
  - skipped
  - pending / todo
- Wahrheitswert (implizit)

Ergebnisse

```
import org.specs2.mutable._

object AnotherFluxCapacitorSpec extends Specification {
  "The flux capacitor" should {
    "be carried by a DeLorean" in {
      FluxCapacitor.carrier must startWith("Trabant")
      FluxCapacitor.carrier must endWith("DMC-12")
    }
  }
}
```

Ergebnisse

```
import org.specs2.mutable._

object AnotherFluxCapacitorSpec extends Specification {
  "The flux capacitor" should {
    "be carried by a DeLorean" in {
      FluxCapacitor.carrier must startWith("Trabant")
      FluxCapacitor.carrier must endWith("DMC-12")
    }
  }
}
```

wirft Exception

Ergebnisse

```
import org.specs2._

object FluxCapacitorSpecFaulty extends Specification { def is =
  "The flux capacitor should be carried by a DeLorean" ! e1

  def e1 = {
    FluxCapacitor.carrier must startWith("Trabant")
    FluxCapacitor.carrier must endWith("DMC-12")
  }
}
```

Ergebnisse

```
import org.specs2._

object FluxCapacitorSpecFaulty extends Specification { def is =
  "The flux capacitor should be carried by a DeLorean" ! e1

  def e1 = {
    FluxCapacitor.carrier must startWith("Trabant")
    FluxCapacitor.carrier must endWith("DMC-12")
  }
}
```

MatchResult[String]
wird verworfen

Ergebnisse

```
import org.specs2._

object FluxCapacitorSpecFaulty extends Specification { def is =
  "The flux capacitor should be carried by a DeLorean" ! e1

  def e1 = {
    FluxCapacitor.carrier must startWith("Trabant")
    FluxCapacitor.carrier must endWith("DMC-12")
  }
}
```

MatchResult[String]
wird verworfen

```
def add(a: Int, b: Int): Int = {
  a * b
  a + b
}
```

Ergebnisse

```
import org.specs2._

object FluxCapacitorSpecRepaired1 extends Specification { def is =
  "The flux capacitor should be carried by a DeLorean" ! e1

  def e1 = FluxCapacitor.carrier must startWith("Trabant") and
                                      endWith("DMC-12")
}
```

Ergebnisse

```
import org.specs2._

object FluxCapacitorSpecRepaired1 extends Specification { def is =
  "The flux capacitor should be carried by a DeLorean" ! e1

  def e1 = FluxCapacitor.carrier must startWith("Trabant") and
                                     endWith("DMC-12")
}
```

Ergebnisse

```
import org.specs2._
import org.specs2.matcher.ThrownExpectations

object FluxCapacitorSpecRepaired2
extends Specification with ThrownExpectations { def is =
  "The flux capacitor should be carried by a DeLorean" ! e1

  def e1 = {
    FluxCapacitor.carrier must startWith("Trabant")
    FluxCapacitor.carrier must endWith("DMC-12")
  }
}
```

Ergebnisse

```
import org.specs2._
import org.specs2.matcher.ThrownExpectations

object FluxCapacitorSpecRepaired2
extends Specification with ThrownExpectations { def is =
  "The flux capacitor should be carried by a DeLorean" ! e1

  def e1 = {
    FluxCapacitor.carrier must startWith("Trabant")
    FluxCapacitor.carrier must endWith("DMC-12")
  }
}
```

Ergebnisse

```
import org.specs2._
import org.specs2.matcher.ThrownExpectations

object FluxCapacitorSpecRepaired2
extends Specification with ThrownExpectations { def is =
  "The flux capacitor should be carried by a DeLorean" ! e1

  def e1 = {
    FluxCapacitor.carrier must startWith("Trabant")
    FluxCapacitor.carrier must endWith("DMC-12")
  }
}
```

wirft Exception

Standard-Ergebnisse

```
import org.specs2.mutable._

object FluxCapacitorSpec extends Specification {
  "The flux capacitor" should {
    "be carried by a DeLorean" >> { todo }

    "require approx. 1.21 gigawatts of electrical power" >> { todo }

    "not yet be able to use the MrFusion power converter" >> { todo }
  }
}
```

Standard-Ergebnisse

```
import org.specs2.mutable._

object FluxCapacitorSpec extends Specification {
  "The flux capacitor" should {
    "be carried by a DeLorean" in {
      FluxCapacitor.carrier must be equalTo "DeLorean DMC-12"
    }

    "require approx. 1.21 gigawatts of electrical power" >> { todo }

    "not yet be able to use the MrFusion power converter" >> { todo }
  }
}
```

Standard-Ergebnisse

```
import org.specs2.mutable._

object FluxCapacitorSpec extends Specification {
  "The flux capacitor" should {
    "be carried by a DeLorean" in {
      FluxCapacitor.carrier must be equalTo "DeLorean DMC-12"
    }

    "require approx. 1.21 gigawatts of electrical power" in {
      FluxCapacitor.requiredPower must be closeTo(1.21e9 +/- 1e6)
    }

    "not yet be able to use the MrFusion power converter" >> { todo }
  }
}
```

Standard-Ergebnisse

```
import org.specs2.mutable._

object FluxCapacitorSpec extends Specification {
  "The flux capacitor" should {
    "be carried by a DeLorean" in {
      FluxCapacitor.carrier must be equalTo "DeLorean DMC-12"
    }

    "require approx. 1.21 gigawatts of electrical power" in {
      FluxCapacitor.requiredPower must be closeTo(1.21e9 +/- 1e6)
    }

    "not yet be able to use the MrFusion power converter" in {
      FluxCapacitor.use(MrFusion) must throwA[SciFiException]
    }
  }
}
```

Standard-Ergebnisse

```
import org.specs2.mutable._

object FluxCapacitorSpec extends Specification {
  "The flux capacitor" should {
    "be carried by a DeLorean" in {
      FluxCapacitor.carrier must be equalTo "DeLorean DMC-12"
    }

    "require approx. 1.21 gigawatts of electrical power" in {
      FluxCapacitor.requiredPower must be closeTo(1.21e9 +/- 1e6)
    }.pendingUntilFixed("ISSUE-42")

    "not yet be able to use the MrFusion power converter" in {
      FluxCapacitor.use(MrFusion) must throwA[SciFiException]
    }
  }
}
```

Automatisches Layout

```
object TimeMachineSpec extends Specification { def is =
  "A time machine should"                                                ^
    "have been invented by Dr. Emmet Brown"                          ! success^
    "break the time barrier at 88 mph"                               ! success^
    "use a flux capacitor that should"                                     ^
      "be carried by a *DeLorean*"                                   ! success^
      "require approx. 1.21 gigawatts of electrical power"           ! success^
      "not yet be able to use the MrFusion power converter"          ! success
}
```

A time machine should
- have been invented by Dr. Emmet Brown
- break the time barrier at 88 mph

use a flux capacitor that should
- be carried by a *DeLorean*
- require approx. 1.21 gigawatts of electrical power
- not yet be able to use the MrFusion power converter

Automatisches Layout

```
object TimeMachineSpec extends Specification { def is =
  "A time machine should"                                                ^
    "have been invented by Dr. Emmet Brown"                          ! success^
    "break the time barrier at 88 mph"                               ! success^
    "use a flux capacitor that should"                                     ^
      "be carried by a *DeLorean*"                                   ! success^
      "require approx. 1.21 gigawatts of electrical power"           ! success^
      "not yet be able to use the MrFusion power converter"          ! success
}
```

markdown

A time machine should
- have been invented by Dr. Emmet Brown
- break the time barrier at 88 mph

use a flux capacitor that should
- be carried by a *DeLorean*
- require approx. 1.21 gigawatts of electrical power
- not yet be able to use the MrFusion power converter

Formatierungs-Fragmente

- nur in Acceptance-Spec

```
object TimeMachineSpecLayout extends Specification { def is =
  "A time machine should"                                                ^
    "have been invented by Dr. Emmet Brown"                    ! success^
    "break the time barrier at 88 mph"                         ! success^
                                                                       br^
    "use a flux capacitor that should"                                  ^
      "be carried by a DeLorean"                               ! success^
      "require approx. 1.21 gigawatts of electrical power"     ! success^
      "not yet be able to use the MrFusion power converter"    ! success^
                                                                    endp^
  "In the end, the time machine is destroyed by an oncoming" +
  "freight train."
}
```

Formatierungs-Fragmente

- nur in Acceptance-Spec

```
object TimeMachineSpecLayout extends Specification { def is =
  "A time machine should"
    "have been invented by Dr. Emmet Brown"
    "break the time barrier at 88 mph"

    "use a flux capacitor that should"
      "be carried by a DeLorean"
      "require approx. 1.21 gigawatts of electrical power"
      "not yet be able to use the MrFusion power converter"

  "In the end, the time machine is destroyed by an oncoming" +
  "freight train."
}
```

```
                                  ^
! success^
! success^
        br^
          ^
! success^
! success^
! success^
    endp^
```

```
object TimeMachineSpecLayout extends Specification { def is =
                                                              args(noindent = true) ^
  "A time machine should"                                                             ^
  "have been invented by Dr. Emmet Brown"                                  ! success^
  "break the time barrier at 88 mph"                                       ! success^
  "use a flux capacitor that should"                                                  ^
  "be carried by a DeLorean"                                               ! success^
  "require approx. 1.21 gigawatts of electrical power"                     ! success^
  "not yet be able to use the MrFusion power converter"                    ! success^
  "In the end, the time machine is destroyed by an " +
  "oncoming freight train."
}
```

```
object TimeMachineSpecLayout extends Specification { def is =
                                                              args(noindent = true) ^
                                                                                    ^
  "A time machine should"                                                           ^
  "have been invented by Dr. Emmet Brown"                                 ! success^
  "break the time barrier at 88 mph"                                      ! success^
  "use a flux capacitor that should"                                                ^
  "be carried by a DeLorean"                                              ! success^
  "require approx. 1.21 gigawatts of electrical power"                    ! success^
  "not yet be able to use the MrFusion power converter"                   ! success^
  "In the end, the time machine is destroyed by an " +
  "oncoming freight train."
}
```

Optionen

- Auswahl
- Ausführung
- Statistiken
- Ausgabe

Optionen

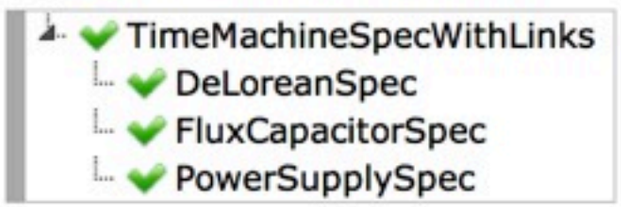
- am Anfang einer Spec
- Kommandozeile
- System-Property
- sbt-Option

Einbinden anderer Specs

```
object TimeMachineSpec extends Specification { def is =
  "A time machine".title                                ^
    DeLoreanSpec                                        ^
    include(FluxCapacitorSpec)                          ^
    include(xonly, PowerSupplySpec)                     ^
    "a regular example"                         ! todo
}
```

Verlinken anderer Specs

```
object TimeMachineSpecWithLinks extends Specification { def is =
  "A time machine consists of"                                    ^
    "a DeLorean" ~ DeLoreanSpec                                   ^
    "the marvellous " ~ ("FluxCapacitor", FluxCapacitorSpec)      ^
    link(PowerSupplySpec)                                         ^
                                                                 p^
    "some history"                                          ! todo
}
```



TimeMachineSpecWithLinks *(issues only)*

A time machine consists of
- a DeLorean
- the marvellous FluxCapacitor
- PowerSupplySpec

- some history
- TODO

Total for specification TimeMachineSpecWithLinks	
Finished in	181 ms
Results	5 examples, 1 failure, 0 error, 1 pending

Copy & Paste

```
object PowerSupplySpecCopies extends Specification { def is =
  "A nuclear reactor power supply should"                                  ^
    "supply at least 1.21 gigawatts of electrical power"          ! e1^
    "fit into a DeLorean DMC-12"                                  ! e2^
    "be named 'Nuclear Reactor'"                                  ! e3^
                                                                  endp^
  "A lightning bolt power supply should"                                   ^
    "supply at least 1.21 gigawatts of electrical power"          ! e4^
    "fit into a DeLorean DMC-12"                                  ! e5^
    "be named 'Lightning Bolt'"                                   ! e6^
                                                                  endp^
  "A Mr Fusion power supply should"                                        ^
    "supply at least 1.21 gigawatts of electrical power"          ! e7^
    "fit into a DeLorean DMC-12"                                  ! e8^
    "be named 'Mr Fusion'"                                        ! e9

  def e1 = NuclearReactor.supply must be greaterThanOrEqualTo 1.21e9
  def e2 = NuclearReactor.volume must be lessThan 150
  def e3 = NuclearReactor.name must be equalTo "Nuclear Reactor"

  def e4 = LightningBolt.supply must be greaterThanOrEqualTo 1.21e9
  def e5 = LightningBolt.volume must be lessThan 150
  def e6 = LightningBolt.name must be equalTo "Lightning Bolt"

  def e7 = MrFusion.supply must be greaterThanOrEqualTo 1.21e9
  def e8 = MrFusion.volume must be lessThan 150
  def e9 = MrFusion.name must be equalTo "Mr Fusion"
}
```

Copy & Paste

```
object PowerSupplySpecCopies extends Specification { def is =
  "A nuclear reactor power supply should"                                  ^
    "supply at least 1.21 gigawatts of electrical power"                  ! e1^
    "fit into a DeLorean DMC-12"                                          ! e2^
    "be named 'Nuclear Reactor'"                                          ! e3^
                                                                          endp^
  "A lightning bolt power supply should"                                   ^
    "supply at least 1.21 gigawatts of electrical power"                  ! e4^
    "fit into a DeLorean DMC-12"                                          ! e5^
    "be named 'Lightning Bolt'"                                           ! e6^
                                                                          endp^
  "A Mr Fusion power supply should"                                        ^
    "supply at least 1.21 gigawatts of electrical power"                  ! e7^
    "fit into a DeLorean DMC-12"                                          ! e8^
    "be named 'Mr Fusion'"                                                ! e9

  def e1 = NuclearReactor.supply must be greaterThanOrEqualTo 1.21e9
  def e2 = NuclearReactor.volume must be lessThan 150
  def e3 = NuclearReactor.name must be equalTo "Nuclear Reactor"

  def e4 = LightningBolt.supply must be greaterThanOrEqualTo 1.21e9
  def e5 = LightningBolt.volume must be lessThan 150
  def e6 = LightningBolt.name must be equalTo "Lightning Bolt"

  def e7 = MrFusion.supply must be greaterThanOrEqualTo 1.21e9
  def e8 = MrFusion.volume must be lessThan 150
  def e9 = MrFusion.name must be equalTo "Mr Fusion"
}
```

Gemeinsame Beispiele

```
object PowerSupplySpecShared extends Specification { def is =
  "A nuclear reactor power supply should"                                    ^
    "behave like any good power supply"                                ^a(NuclearReactor)^
    "be named '${Nuclear Reactor}'"                                 ! equalName(NuclearReactor)^
                                                                            endp^
  "A lightning bolt power supply should"                                     ^
    "behave like any good power supply"                                ^a(LightningBolt)^
    "be named '${Lightning Bolt}'"                                  ! equalName(LightningBolt)^
                                                                            endp^
  "A Mr Fusion power supply should"                                          ^
    "behave like any good power supply"                                    ^a(MrFusion)^
    "be named '${Mr Fusion}'"                                        ! equalName(MrFusion)

  def a(p: => PowerSupply) =
    "supply at least 1.21 gigawatts of electrical power"  ! Any(p).e1^
    "fit into a DeLorean DMC-12"                          ! Any(p).e2

  case class Any(p: PowerSupply) {
    def e1 = p.supply must be greaterThanOrEqualTo 1.21e9
    def e2 = p.volume must be lessThan 150
  }

  def equalName(p: PowerSupply) = so { case n: String => p.name must be equalTo n }
}
```

Gemeinsame Beispiele

```
object PowerSupplySpecShared extends Specification { def is =
  "A nuclear reactor power supply should"                                    ^
    "behave like any good power supply"                                  ^a(NuclearReactor)^
    "be named '${Nuclear Reactor}'"                                ! equalName(NuclearReactor)^
                                                                           endp^
  "A lightning bolt power supply should"                                     ^
    "behave like any good power supply"                                  ^a(LightningBolt)^
    "be named '${Lightning Bolt}'"                                 ! equalName(LightningBolt)^
                                                                           endp^
  "A Mr Fusion power supply should"                                          ^
    "behave like any good power supply"                                    ^a(MrFusion)^
    "be named '${Mr Fusion}'"                                      ! equalName(MrFusion)

  def a(p: => PowerSupply) =
    "supply at least 1.21 gigawatts of electrical power"  ! Any(p).e1^
    "fit into a DeLorean DMC-12"                          ! Any(p).e2

  case class Any(p: PowerSupply) {
    def e1 = p.supply must be greaterThanOrEqualTo 1.21e9
    def e2 = p.volume must be lessThan 150
  }

  def equalName(p: PowerSupply) = so { case n: String => p.name must be equalTo n }
}
```

Gemeinsame Beispiele

```
object PowerSupplySpecShared extends Specification { def is =
  "A nuclear reactor power supply should"                                    ^
    "behave like any good power supply"                                ^a(NuclearReactor)^
    "be named '${Nuclear Reactor}'"                                ! equalName(NuclearReactor)^
                                                                           endp^
  "A lightning bolt power supply should"                                     ^
    "behave like any good power supply"                                ^a(LightningBolt)^
    "be named '${Lightning Bolt}'"                                 ! equalName(LightningBolt)^
                                                                           endp^
  "A Mr Fusion power supply should"                                          ^
    "behave like any good power supply"                                    ^a(MrFusion)^
    "be named '${Mr Fusion}'"                                      ! equalName(MrFusion)

  def a(p: => PowerSupply) =
    "supply at least 1.21 gigawatts of electrical power"  ! Any(p).e1^
    "fit into a DeLorean DMC-12"                          ! Any(p).e2

  case class Any(p: PowerSupply) {
    def e1 = p.supply must be greaterThanOrEqualTo 1.21e9
    def e2 = p.volume must be lessThan 150
  }

  def equalName(p: PowerSupply) = so { case n: String => p.name must be equalTo n }
}
```

Automatischer Beispieltext

```
object PowerSupplySpecAutoExample extends Specification { def is =
  "A nuclear reactor power supply should"                                    ^
    "behave like any good power supply"                          ^ a(NuclearReactor)^
    { NuclearReactor.name should be equalTo "Nuclear Reactor" }                ^
                                                                         endp^
  "A lightning bolt power supply should"                                     ^
    "behave like any good power supply"                           ^ a(LightningBolt)^
    { LightningBolt.name should be equalTo "Lightning Bolt" }                  ^
                                                                         endp^
  "A Mr Fusion power supply should"                                          ^
    "behave like any good power supply"                           ^ a(MrFusion)^
    { MrFusion.name should be equalTo "Mr Fusion" }                          ^end

  def a(p: => PowerSupply) =
    "supply at least 1.21 gigawatts of electrical power"           ! Any(p).e1^
    "fit into a DeLorean DMC-12"                                   ! Any(p).e2

  case class Any(p: PowerSupply) {
    def e1 = p.supply must be greaterThanOrEqualTo 1.21e9
    def e2 = p.volume must be lessThan 150
  }
}
```

Given-When-Then

```
object InventorSpec extends Specification { def is =
  "Given the inventor ${Dr. Emmet Brown}"    ^ inventor  ^
  "when in the year ${2015}"                 ^ year      ^
  "then he should invent ${Mr Fusion}"       ^ invention

  def inventor = new Given[Inventor] {
    def extract(text: String): Inventor = new Inventor(extract1(text))
  }

  def year = new When[Inventor, PowerSupply] {
    def extract(doc: Inventor, text: String) =
      doc.inventPowerSupplyInTheYear(extract1(text).toInt)
  }

  def invention = new Then[PowerSupply] {
    def extract(invention: PowerSupply, text: String) =
      invention.name must be equalTo extract1(text)
  }
}
```

Kontext

```
import org.specs2.mutable._

object TimeMachineSpecContext extends Specification {
  "A time machine" should {
    "change the current year" in new movie {
      travel.currentYear must be equalTo 1955
    }
  }

  trait movie extends After {
    lazy val travel = new TimeMachine

    def after = travel.backToTheFuture()
  }
}
```

Context

- Before
- After
- Around
- Outside
- Step
- Action

Weiteres

- Titel
- Etiketten (tags)
- Abschnitte (sections)
- Index-Seite

behaviour-driven

fly-by

setup & running

structure

matchers

roundup

Standard-Typen

```
42 must be equalTo 42

"ABBA" must beMatching("[AB]{4}")

1 / 0 must throwAn[ArithmeticException]

List(1, 2, 3) must have size 3

1.5 must be closeTo(1.0 +/- 0.5)

<a><b><c attr="value"/></b></a> must \\("c", "attr" -> "value")

"/dev/null" must beAWritablePath
```

Varianten

```
42 must beEqualTo(42)
42 must be_==(42)
42 must_== 42
42 mustEqual 42
42 should_== 42
42 === 42
42 must be equalTo 42

23 must not be equalTo(2)
23 must_!= 42
23 mustNotEqual 42
23 must be_!=(42)
23 !== 42
```

Beschreibung

```
FluxCapacitor.inventor aka
  "the inventor of the flux capacitor" must
  be equalTo "Dr. Emmet Brown"
```

```
"TwoWords" as camel must be equalTo "TwoWords"

def camel(s: String) = s.replaceAll("([A-Z])", " $1").toLowerCase.trim
```

Auflistungen

```
{ (_:Int) must be lessThan 23 } foreach List(1, 2, 3, 4, 5)

{ (_:Int) must be lessThan 23 } forall List(1, 2, 3, 4, 5)

{ (_:Int) must be lessThan 23 } atLeastOnce List(1, 2, 3, 4, 5)
```

Eigene Matcher

```
object MatchersSpec extends Specification { def is =
  "matchers should match" ! {

    val iter = List(1, 2, 3) iterator

    iter.next must eventuallyBe(3)
  }

  val eventuallyBe = beEqualTo(_:Int).eventually
}
```

Eigene Matcher

```
object MatchersSpec extends Specification { def is =
  "matchers should match" ! {

    3 must beBetween(1, 5)

  }

  def beBetween(a: Int, b: Int) = be_>=(a) and be_<=(b)
}
```

Eigene Matcher

```
object MatchersSpec extends Specification { def is =
  "matchers should match" ! {

    List(1, 2, 3) must haveAShortStringRepresentation

  }

  def haveAShortStringRepresentation =
    beLessThanOrEqualTo(10) ^^ { (a: Any) => a.toString.size }
}
```

Eigene Matcher

```
object MatchersSpec extends Specification { def is =
  "matchers should match" ! {

    "fear" must beAnFWord

  }

  val beAnFWord: Matcher[String] =
    ((_: String).toLowerCase.startsWith("f"), "doesn't start with f")
}
```

Mocks

```
object MockingSpec extends Specification {
  "The road runner" should {
    "run the road after meeping" in new canyon {
      roadRunner meepAt "Wile E. Coyote"
      there was atLeastOne(road).run
    }
  }

  trait canyon extends Scope with Mockito {
    lazy val road = mock[Runnable]
    lazy val roadRunner = new RoadRunner(road)
  }
}

class RoadRunner(road: Runnable) {
  def meepAt(target: String) = {
    println("Meep meep, %s!" format target)
    road.run
  }
}
```

Testfall-Tabellen

```
import org.specs2._
import org.specs2.matcher.DataTables

object DataTableSpec extends Specification with DataTables { def is =
  "A string should be found inside a phrase" ! example

  def example =
    "phrase"                      || "string" || "expected position" |
    "Hello world!"                !! "Hell"   !! 0                   |
    "The quick brown fox..."      !! "own"    !! 12                  |
    "Somewhere over the rainbow" !! "misery" !! -1                  |> {

    (phrase, string, position) => phrase indexOf string must be equalTo position

  }
}
```

Testfall-Tabellen

```
import org.specs2._
import org.specs2.matcher.DataTables

object DataTableSpec extends Specification with DataTables { def is =
  "A string should be found inside a phrase" ! example

  def example =
    "phrase"                          || "string" || "expected position" |
    "Hello world!"                    !! "Hell"   !! 0                   |
    "The quick brown fox..."          !! "own"    !! 12                  |
    "Somewhere over the rainbow"      !! "misery" !! -1                  |> {

    (phrase, string, position) => phrase indexOf string must be equalTo position

  }
}
```

Testfall-Tabellen

```
import org.specs2._
import org.specs2.matcher.DataTables

object DataTableSpec extends Specification with DataTables { def is =
  "A string should be found inside a phrase" ! example

  def example =
    "phrase"                          || "string" || "expected position" |
    "Hello world!"                    !! "Hell"   !! 0                   |
    "The quick brown fox..."          !! "own"    !! 12                  |
    "Somewhere over the rainbow"      !! "misery" !! -1                  |> {

    (phrase, string, position) => phrase indexOf string must be equalTo position

  }
}
```

Testfall-Tabellen (Erfolg)

A string should be found inside a phrase

phrase	string	expected position
Hello world!	Hell	0
The quick brown fox...	own	12
Somewhere over the rainbow	misery	-1

Testfall-Tabellen (Fehlschlag)

⚠ A string should be found inside a phrase

phrase	string	expected position	message
Hello world!	Hell	1	'0' is not equal to '1'
The quick brown fox...	own	42	'12' is not equal to '42'
Somewhere over the rainbow	misery	-1	

Formulare (forms)

- Datenstrukturen
- Geschäftsabläufe
- tabellarische Form

Entscheidungstabelle

a	b	a + b	a – b
2	1	3	1
2	2	4	0
23	42	65	-19

Entscheidungstabelle

```
object Calculator {
  def add(a: Int, b: Int) = a + b
  def subtract(a: Int, b: Int) = math.abs(a - b)
}
```

Entscheidungstabelle

```
object CalculatorSpec extends Specification { def is =
  "A calculator should add and subtract correctly" ^
    CalculatorForm(Form.th("a", "b", "a + b", "a - b")).
                       tr(  2,   1,    3,        1    ).
                       tr(  2,   2,    4,        0    ).
                       tr( 23,  42,   65,      -19    )
}

case class CalculatorForm(form: Form) {
  def tr(a: Int, b: Int, a_plus_b: Int, a_minus_b: Int) =
    CalculatorForm(form.tr(
      a,
      b,
      prop( Calculator.add(a, b)      )(a_plus_b),
      prop( Calculator.subtract(a, b) )(a_minus_b)))
}
```

Entscheidungstabelle

CalculatorSpec *(issues only)*

A calculator should add and subtract correctly

a	b	a + b	a - b	
2	1	3	1	
2	2	4	0	
23	42	65	-19	'19' is not equal to '-19'

[click on failed cells to see the stacktraces]

ScalaCheck

- Property-Based Testing
- Invarianten
- viele zufällige Testfälle

ScalaCheck

```
object SquareRootSpec extends Specification with ScalaCheck {
  "The integer square root of n" should {
    "never be negative" in check {
      (n: Int) => SquareRoot of n should be greaterThanOrEqualTo 0
    }
  }
}
```

ScalaCheck

The integer square root of n should
never be negative
A counter-example is '-1': java.lang.ArithmeticException: n must be >= 0 (after 1 try)
(SquareRootSpec.scala:12)

ScalaCheck

```
object SquareRootSpec extends Specification with ScalaCheck {
  "never be negative" in check {
    (n: Int) => (n >= 0) ==> {
      SquareRoot of n should be greaterThanOrEqualTo 0
    }
  }
}
```

ScalaCheck

```
object SquareRootSpec extends Specification with ScalaCheck {
  "never be negative" in check {
    (n: Int) => (n >= 0) ==> {
      SquareRoot of n should be greaterThanOrEqualTo 0
    }
  }
}
```

ScalaCheck

```
object SquareRootSpec extends Specification with ScalaCheck {
  "The integer square root of n" should {
    "never be negative" in check { positiveInteger {
      (n: Int) => SquareRoot of n should be greaterThanOrEqualTo 0
    }}
  }

  def positiveInteger = Arbitrary {
    for (n <- arbitrary[Int]) yield
     if (n == Int.MinValue) Int.MaxValue
     else math.abs(n)
  }
}
```

ScalaCheck

```
object SquareRootSpec extends Specification with ScalaCheck {
  "The integer square root of n" should {
    "never be negative" in check { positiveInteger {
      (n: Int) => SquareRoot of n should be greaterThanOrEqualTo 0
    }}
  }

  def positiveInteger = Arbitrary {
    for (n <- arbitrary[Int]) yield
     if (n == Int.MinValue) Int.MaxValue
     else math.abs(n)
  }
}
```

ScalaCheck

```
object SquareRootSpec extends Specification with ScalaCheck {
  "The integer square root of n" should {
    "never be negative" in check { positiveInteger {
      (n: Int) => SquareRoot of n should be greaterThanOrEqualTo 0
    }.set(minTestsOk -> 500)}
  }

  def positiveInteger = Arbitrary {
    for (n <- arbitrary[Int]) yield
     if (n == Int.MinValue) Int.MaxValue
     else math.abs(n)
  }
}
```

ScalaCheck

```
object SquareRootSpec extends Specification with ScalaCheck {
  "The integer square root of n" should {
    "never be negative" in check { positiveInteger {
      (n: Int) => SquareRoot of n should be greaterThanOrEqualTo 0
    }.set(minTestsOk -> 500)}
  }

  def positiveInteger = Arbitrary {
    for (n <- arbitrary[Int]) yield
     if (n == Int.MinValue) Int.MaxValue
     else math.abs(n)
  }
}
```

behaviour-driven

fly-by

setup & running

structure

matchers

roundup

specs2

- ausdrucksstarke DSL
- gute Tool-Anbindung
- Open Source (specs2.org)

Andere Bibliotheken

- ScalaTest (morgen 11:20)
- JUnit / TestNG + FEST + Mockito
- ausprobieren!

„The first step is just to try to write readable and maintainable specs. The rest follows.“

Eric Torreborre

5.– 8. September 2011
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Vielen Dank

Andreas Flierl
imbus AG

imbus AG

- Spezialisierter Lösungsanbieter für Software-Qualitätssicherung und Software-Test
- Innovativ seit 1992
- Erfahrung und Know-how aus über 3.000 erfolgreichen Projekten
- 170 Mitarbeiter an vier Standorten in Deutschland
- Beratung, Test-Services, Training, Tools, Datenqualität
- Für den gesamten Software-Lebenszyklus

www.imbus.de