

# Jnario

Executable Specifications for Java

Sebastian Benz - E.S.R. Labs

Birgit Engelmann - BMW Car IT GmbH

```
public class CustomerLookupTest{

    @Test
    public void findsCustomerById() { ...
    }
    @Test
    public void failsForDuplicateCustomers() { ...
    }

    ...
}
```

CustomerLookup

findsCustomerById

failsForDuplicateCustomers

...

## Customer Lookup

- finds Customer by Id
- fails for Duplicate Customers
- ...

Executable Specification

```
public class ResourceSetTest {  
  
    @Test  
    public void loadingResourcesByFileUri() {  
        givenAnEcoreFile("Example.ecore");  
  
        URI fileUri = URI.createFileURI("Example.ecore");  
        Resource resource = new ResourceSetImpl().getResource(fileUri);  
  
        assertThat(resource.getContents(), is(not(empty())));  
    }  
  
}
```

# ResourceSet

Loading Resources by File Uri:

```
givenAnEcoreFile("Example.ecore");  
  
URI fileUri = URI.createFileURI("Example.ecore");  
Resource resource = new ResourceSetImpl().createResource(fileUri);  
  
assertThat(resource.getContents(), is(not(empty())));
```



Executable Documentation

Tests can be:

- executable specifications
- executable documentation

...with JUnit?

# Jnario

- A DSL for testing based on Xtend
- Readable acceptance specifications
- Succinct unit tests
- Executable documentation



# Demo: Coffee Tracker

# Cucumber-style Specs with a Twist

 Calculator.feature ✕

package calculator

## Feature: Addition

In order to avoid silly mistakes  
As a math idiot  
I want to be told the sum of two numbers

### Scenario: Add two numbers

**When** I entered "50" and "70"  
**Then** the result should be "120"

# Cucumber-style Specs with a Twist

Calculator.feature ✕

```
package calculator
```

## Feature: Addition

In order to avoid silly mistakes

As a math idiot

I want to be told the sum of two numbers

## Scenario: Add two numbers

```
val calculator = new Calculator()
```

```
int result
```

**When** I entered "50" and "70"

```
result = calculator.add(args.first, args.second)
```

**Then** the result should be "120"

```
result => args.first.toInt
```

# Write succinct Unit Specifications

Stack.spec ✕

```
describe Stack{  
  context "empty"{  
    fact subject.empty should be true  
    fact subject.pop() throws EmptyStackException  
  }  
  context "not empty"{  
    fact "increases size when pushing"{  
      subject.push("something")  
      subject.size => 1  
    }  
    fact "decreases size when popping"{  
      subject.push("something")  
      subject.pop()  
      subject.size => 0  
    }  
  }  
}
```

# 100% Java & JUnit compatible

The image shows a screenshot of the JUnit test runner interface. At the top, it says "JUnit" with a green checkmark icon. Below that, it states "Finished after 0.1 seconds". The summary bar shows "Runs: 60/60", "Errors: 0", and "Failures: 0". A green progress bar is visible below the summary. The test results are listed in a tree view:

- ▼ Jnario Examples [Runner: JUnit 4] (0.056 s)
  - ▼ Calculator (0.000 s)
    - ▼ Addition (0.000 s)
      - ▶ Scenario: Add two numbers (0.000 s)
  - ▼ Game of Life (0.056 s)
    - ▼ Game of Life with Jnario & Xtend (0.056 s)
      - ▶ Features (0.021 s)
      - ▼ Unit Specifications (0.035 s)
        - ▶ CellLocation (0.005 s)
        - ▶ Evolution (0.007 s)
        - ▶ Rules (0.006 s)
        - ▶ Select multiple specs
        - ▶ World (0.012 s)
- ▶ Diverse



# Test Suites on Steroids

Example4.suite ✕

## #Test Suites on Steroids

Describe your test suite in plain **\*\*formatted\*\*** text.

### ##List existing Specs

Reference existing specs.

- **"Stack"**: this is a reference to our Stack specification.

### ##Select multiple specs

Select multiple specs using regular expressions.

- **\diverse.\*\**

# Executable Documentation

\*XtendFacts.spec

```
/*
 * This document teaches you everything
 * you need to know about Xtend to effectively use
 * [Jnario](http://www.jnario.org). For a more detailed
 * introduction see the official [Xtend documentation]
 * (http://www.eclipse.org/xtend/documentation.html).
 * This document is automatically generated from a specification
 * written in Jnario. You can see the original source code
 * by clicking on **source** in the upper right corner.
 */
describe "20 Facts about Xtend"{
  /*
   * The syntax of Xtend is quite similar to Java, but Xtend code
   * is usually a lot shorter than its Java counterpart. However,
   * there are some important differences between Java and Xtend
   * one should be aware of.
   */
  describe "A modernized Java"{
    /*
     * Semicolons are optional in Xtend.
     */
    fact "No semicolons"{
      var greeting = ""
      greeting = "Hello World"
      println(greeting)
    }
    /*
     * Variable declarations are preceded by `var` or, in case of
     * final variables, by `val`.
     */
    fact "Variables are declared with var and val"{
      var String x = "I might change"
      val String y = "I'll never change" // final
    }
  }
}
```

# Executable Documentation

\*XtendFacts.spec

```
/*
 * This document teaches you
 * you need to know about Xtend
 * [Jnario](http://www.jnario.org)
 * introduction see the official
 * (http://www.eclipse.org/xtend/)
 * This document is automatically
 * written in Jnario. You can see
 * by clicking on **source**
 */
describe "20 Facts about Xtend" {
  /*
   * The syntax of Xtend is
   * is usually a lot shorter than
   * there are some important
   * one should be aware of
   */
  describe "A modernized Java" {
    /*
     * Semicolons are optional
     */
    fact "No semicolons" {
      var greeting = ""
      greeting = "Hello World"
      println(greeting)
    }
    /*
     * Variable declaration
     * final variables, by
     */
    fact "Variables are declared with var and val" {
      var String x = "I might change"
      val String y = "I'll never change"
    }
  }
}
```

## 20 Facts about Xtend

Spec

Source

This document teaches you everything you need to know about Xtend to effectively use [Jnario](#). For a more detailed introduction see the official [Xtend documentation](#). This document is automatically generated from a specification written in Jnario. You can see the original source code by clicking on **source** in the upper right corner.

### A modernized Java

The syntax of Xtend is quite similar to Java, but Xtend code is usually a lot shorter than its Java counterpart. However, there are some important differences between Java and Xtend one should be aware of.

- **No semicolons**

Semicolons are optional in Xtend.

```
1. var greeting = ""
2. greeting = "Hello World"
3. println(greeting)
```

- **Variables are declared with var and val**

Variable declarations are preceded by **var** or, in case of final variables, by **val**.

```
1. var String x = "I might change"
2. val String y = "I'll never change" // final
```



# www.jnario.org

Sebastian Benz - @sebabenzenz

Brigit Engelmann - @engelmannb