

Unit Testing

Introduction

❖ We can use various tools for performing our unit testing.

These tools can be either Java based tools (such as `JUnit` or `TestNG`) or Scala based tools (such as `ScalaTest`, `Specs` and `ScalaCheck`).

Introduction

The screenshot shows a web browser window displaying the JUnit.org website. The browser's address bar shows the URL <http://www.junit.org/>. The website has a green header with the text "JUnit.org Resources for Test Driven Development" and a navigation menu with links for Home, Download JUnit, JavaDoc, Project Homepage, Getting Started, and About.

Navigation

- JUnit Releases
- Issue Tracker
- News
 - Articles
 - Announcements
- Mailing List
- Web Forums
- Tools
- Submit Content

Welcome to JUnit.org!


This site is dedicated to software developers and testers using the JUnit testing framework. In addition to the latest news and download links for JUnit, you can find answers to questions about automated testing, tools and extensions for JUnit, and articles on Test Driven Development and other testing topics.

Our goal is to serve you. We've redesigned the site to better meet this goal. If you have questions or suggestions please [contact us](#).

What's New?

- Time flies with tempus-fugit
- Creating abstract contract tests in JUnit 4
- JUnit 4.8.1 is released
- JUnit 4.8 is released
- Writing your own JUnit extensions using @Rule

[more](#)

Syndicate 

User login

Username: *

Password: *

- Create new account
- Request new password

The Windows taskbar at the bottom shows the Start button, several open windows (including Notepad and Internet Explorer), and the system clock displaying 2:01 PM.

Introduction



TestNG

Now available



[Click for more details.](#)

*Cedric Beust (cedric at beust.com)
Current version: 5.11
Created: April 27th, 2004
Last Modified: January 23rd, 2010*

TestNG is a testing framework inspired from JUnit and NUnit but introducing some new functionalities that make it more powerful and easier to use, such as:

- Annotations.
- Flexible test configuration.
- Support for data-driven testing (with `@DataProvider`).
- Support for parameters.
- Allows distribution of tests on slave machines.
- Powerful execution model (no more `TestSuite`).
- Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven, etc...).
- Embeds BeanShell for further flexibility.
- Default JDK functions for runtime and logging (no dependencies).
- Dependent methods for application server testing.

TestNG is designed to cover all categories of tests: unit, functional, end-to-end, integration, etc...

Introduction



The screenshot shows a web browser window displaying the ScalaTest homepage. The browser's address bar shows the URL `http://www.scalatest.org/`. The page features a navigation menu with links for [Home](#), [Download](#), [Quick Start](#), [ScalaDoc](#), [Community](#), and [About](#). The main heading is "ScalaTest™" in red and green, followed by the tagline "Less code, more clarity™". Below this, a paragraph describes ScalaTest as an open-source test framework for the Java Platform designed to increase productivity by letting users write fewer lines of test code that more clearly reveal their intent. A code block in a dark blue box shows Scala code for a `StackSpec` class. The code includes imports for `FlatSpec` and `ShouldMatchers`, and defines two test scenarios: one for a stack of integers and another for a stack of strings that should throw a `NoSuchElementException` when popped. At the bottom of the browser window, the Windows taskbar is visible, showing the Start button and several open applications including Notepad and Internet Explorer. The system clock in the bottom right corner indicates the time is 2:03 PM.

```
import org.scalatest.FlatSpec
import org.scalatest.matchers.ShouldMatchers

class StackSpec extends FlatSpec with ShouldMatchers {

  "A Stack" should "pop values in last-in-first-out order" in {
    val stack = new Stack[Int]
    stack.push(1)
    stack.push(2)
    stack.pop() should equal (2)
    stack.pop() should equal (1)
  }

  it should "throw NoSuchElementException if an empty stack is popped" in {
    val emptyStack = new Stack[String]
    evaluating { emptyStack.pop() } should produce [NoSuchElementException]
  }
}
```

With ScalaTest, you can test either Scala or Java code. By integrating with popular tools such as JUnit, TestNG, Ant, and Maven, ScalaTest makes it easy to take your

Introduction

The screenshot shows a web browser window displaying the 'specs' project page on code.google.com. The browser's address bar shows the URL 'http://code.google.com/p/specs/'. The page header includes the user 'Haim.Michael@gmail.com' and navigation links for 'My favorites', 'Profile', and 'Sign out'. The main content area features the 'specs' logo, a search bar, and a navigation menu with 'Project Home', 'Downloads', 'Wiki', 'Issues', and 'Source'. Below the navigation is a 'Welcome to specs!' section followed by a table of updates. The table lists versions from 1.4.2 to 1.6.1 with their respective release dates and descriptions of new features. To the right of the table are several informational boxes: 'Star this project', 'Code license: MIT License', 'Labels: scala, bdd, junit, scalacheck, scalatest, jmock', 'Featured downloads: scaladocs-1.6.1.zip, specs-1.6.1.jar', 'Featured wiki pages: Changelog, QuickStart, Sponsors, Troubleshooting, UserGuide', 'Links: Scaladoc for specs, Hudson builds', 'Blogs: Eric Torreborre's blog', 'Feeds: Project feeds', and 'Groups: specs user group'. At the bottom of the page, there is a code snippet for a Scala specification.

11/05/2009	specs 1.6.1	NEW: Added "around" actions that can be executed around the expectations of an example. Added SpecContexts . Added a plan option to display the plan of a specification without executing the examples. Other improvements and fixes
09/08/2009	specs 1.6.0	NEW: new execution model with automated clean-up of local variables and first-class subexamples. Added an EasyMock trait . Other improvements and fixes
05/07/2009	specs 1.5.0	NEW: Easier syntax with be/have + matcher . Alpha version of literate specifications and Forms . Other improvements and fixes : run options, configuration, pending examples.
04/06/2009	specs 1.4.4	NEW: Added a Mockito trait to use the Mockito library for mock objects. Added new command-line options . Documented the System contexts and added a way to use them with implicit vals. Changed the Maven group id to org.scala-tools.testing. See fixes and other improvements here
02/12/2009	specs 1.4.3	NEW: Added a "-xonly" or "-failedonly" option to only display failures and errors in the console. Fixed issues and added 2 matchers .
01/14/2009	specs 1.4.2	NEW: Upgraded to scala-2.7.3 . Added a new TeamCity runner (thanks to Stepan Koltsov).

specs is a [Behaviour-Driven-Design](#) framework which provides:

- a simple and typed language to create specifications ([your first specification in 5 minutes](#))

```
object helloWorld extends Specification {
  "hello world" has 11 characters in {
    "hello world".size must be equalTo(11)
  }
}
```

Introduction

The screenshot shows a web browser window displaying the ScalaCheck project page on code.google.com. The browser's address bar shows the URL `http://code.google.com/p/scalacheck/`. The page header includes the user `Haim.Michael@gmail.com` and navigation links for `My favorites`, `Profile`, and `Sign out`. The main content area features the project logo, a search bar, and navigation tabs for `Project Home`, `Downloads`, `Wiki`, `Issues`, and `Source`. Below these are links for `Summary`, `Updates`, and `People`. The main text describes ScalaCheck as a powerful tool for automatic unit testing of Scala and Java programs, highlighting its features like automatic test case generation and minimization. It mentions the author, Rickard Nilsson, and provides a list of news items from 2009. An introduction section explains where to find user guides and API documentation, and a quick start section shows a code snippet for specifying methods of the `java.util.String` class. The right sidebar contains various project details, including a star button, code license (New BSD License), labels (Scala, Testing, QuickCheck, specs, scalatest, scalacheck), featured wiki pages (ChangeHistory, ScalaCheckArticles), links (The Scala Programming Language, QuickCheck, ScalaCheck at Ohloh, Bill Venners' short ScalaCheck presentation at JavaPosse), feeds (Project feeds), groups (ScalaCheck user mailinglist), project owners (rickynils), and project committers (tonymorris, paul.phillips). The Windows taskbar at the bottom shows several open applications and the system clock at 2:08 PM.

scalacheck
A powerful tool for automatic unit testing

Project Home Downloads Wiki Issues Source
Summary | Updates | People

ScalaCheck is a powerful tool for automatic unit testing of Scala and Java programs. It features automatic test case generation and minimization of failing test cases. ScalaCheck started out as a Scala port of the Haskell library [QuickCheck](#), and has since evolved and been extended with features not found in Haskell QuickCheck.

Author: Rickard Nilsson

News

- 2009-11-08 - [New ScalaCheck builds](#)
- 2009-10-14 - [New ScalaCheck builds available](#)
- 2009-10-07 - [ScalaCheck 1.6 released](#)

Introduction

See the [UserGuide](#) for detailed information on ScalaCheck usage, and [ScalaCheck API documentation](#) for reference information. [ScalaCheckArticles](#) provides links to some articles and blog posts about ScalaCheck. There is also a mailing list for users of ScalaCheck, <http://groups.google.com/group/scalacheck>. ScalaCheck is released under a BSD-like [license](#).

Quick Start

Specify some methods of the `java.util.String` class with ScalaCheck:

```
import org.scalacheck._

object StringSpecification extends Properties("String") {
  property("startsWith") = Prop.forAll((a: String, b: String) => (a+b).startsWith(a))

  property("endsWith") = Prop.forAll((a: String, b: String) => (a+b).endsWith(b))

  // Is this really always true?
```

The FunSuite Class

- ❖ The simplest way of using the ScalaTest framework is by defining a new class that extends `FunSuite` and specify the tests within it.

The FunSuite Class

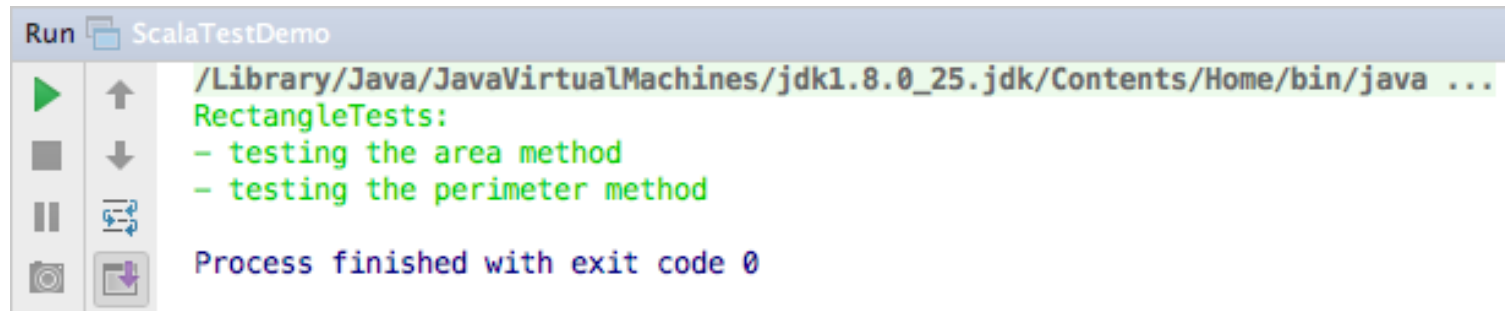
```
object ScalaTestDemo {
  def main(args:Array[String]):Unit = {
    new RectangleTests().execute
  }
}

class Rectangle(private var width:Double, private var height:Double) {
  def area():Double = width*height
  def perimeter():Double = 2*(width+height)
}

class RectangleTests extends FunSuite {

  test("testing the area method") {
    var ob = new Rectangle(3,4)
    assert(ob.area==12)
  }
  test("testing the perimeter method") {
    var ob = new Rectangle(3,4)
    assert(ob.perimeter==14)
  }
}
```

The FunSuite Class



The screenshot shows a console window titled "Run ScalaTestDemo". On the left side, there is a vertical toolbar with icons for play, stop, pause, and refresh. The main area of the console displays the following text:

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home/bin/java ...  
RectangleTests:  
- testing the area method  
- testing the perimeter method  
  
Process finished with exit code 0
```