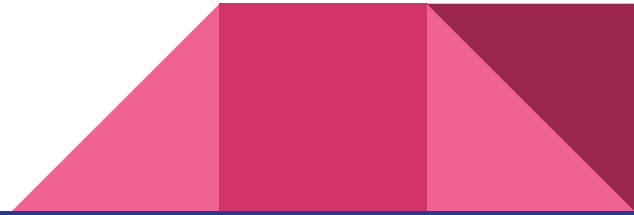
ScalaTest

cerenode.io

Overview

- The central concept in ScalaTest is the *suite*, a collection of zero to many tests
- A *test* can be anything with a name that can start and either succeed, fail, be pending, or canceled
- Trait Suite declares run and other “lifecycle” methods that define a default way to write and run tests
- You define test classes by composing Suite style and mixin traits
- You define test suites by composing Suite instances



Using ScalaTest

- Add the following dependency to your ‘build.sbt’ file

```
libraryDependencies += "org.scalatest" % "scalatest_2.12" % "3.0.5" % "test"
```

Unit Testing

- Testing single unit of a code, like a class
- Smallest part of a project that can be tested
- Purpose is to validate each part of a program works as expected

Integration Testing

- Individual units are combined and tested together as a group
- Used to expose faults in interaction between integrated units
- Second level of testing after unit testing

Acceptance Testing

- System is tested for acceptability
- Evaluate the system’s compliance with business requirements

Testing Styles

- Supports different styles of testing
- Usually implemented by having one main style for unit testing and another for acceptance testing
  - FlatSpec for unit and integration testing
  - FeatureSpec for acceptance testing

Assertions

- Makes 3 assertions by default in any style trait
  - `assert` - for general assertions
  - `assertResult` - to differentiate expected from actual values
  - `assertThrows` - to ensure a bit of code throws an expected exception
- Assertions are defined in `Assertions` trait
- If passed expression is true assert will return normally
- If false it will complete abruptly with an AssertionError

Matchers

- It is a DSL
- Used for expressing assertions in tests using the word `should`
- Provides five different ways to check equality