TESTING SCALA IN INTELLIJ WITH SCALATEST

There are multiple libraries and testing methodologies for Scala, but in this tutorial, we'll demonstrate one popular option from the ScalaTest framework called FunSuite.

This assumes you know how to build a project in IntelliJ.

Setup

1. Create an sbt project in IntelliJ.
2. Add the ScalaTest dependency:
   1. Add the ScalaTest dependency to your `build.sbt` file:

      ```
      libraryDependencies += "org.scalatest" %% "scalatest" % "3.2.11" % Te
      ```

   2. If you get a notification "build.sbt was changed", select **auto-import**.
   3. These two actions will cause `sbt` to download the ScalaTest library.
   4. Wait for the `sbt` sync to finish; otherwise, `AnyFunSuite` and `test()` will be unrecognized.
3. On the project pane on the left, expand `src` => `main`.
4. Right-click on `scala` and select **New** => **Scala class**.
5. Call it `CubeCalculator`, change the **Kind** to `object`, and hit enter or double-click on `object`.
6. Replace the code with the following:

```
object CubeCalculator:
  def cube(x: Int) =
    x * x * x
```

Creating a test

1. On the project pane on the left, expand `src` => `test`.
2. Right-click on `scala` and select **New** => **Scala class**.
3. Name the class `CubeCalculatorTest` and hit enter or double-click on `class`.
4. Replace the code with the following:

```
import org.scalatest.funsuite.AnyFunSuite

class CubeCalculatorTest extends AnyFunSuite:
  test("CubeCalculator.cube") {
    assert(CubeCalculator.cube(3) === 27)
  }
```

5. In the source code, right-click `CubeCalculatorTest` and select **Run 'CubeCalculatorTest'**.

Understanding the code

Let's go over this line by line:

- `class CubeCalculatorTest` means we are testing the object `CubeCalculator`
- `extends AnyFunSuite` lets us use functionality of ScalaTest's AnyFunSuite class such as the `test` function
- `test` is a function that comes from the FunSuite library that collects results from assertions within the function body.
- `"CubeCalculator.cube"` is a name for the test. You can call it anything but one convention is "ClassName.methodName".
- `assert` takes a boolean condition and determines whether the test passes or fails.
- `CubeCalculator.cube(3) === 27` checks whether the output of the `cube` function is indeed 27. The `===` is part of ScalaTest and provides clean error messages.

Adding another test case

1. Add another `assert` statement after the first one that checks for the cube of `0`.
2. Re-run the test again by right-clicking `CubeCalculatorTest` and selecting 'Run **CubeCalculatorTest**'.

Conclusion

You've seen one way to test your Scala code. You can learn more about ScalaTest's FunSuite on the official website.