Unit testing

with Junit5

Software testing

- Unit testing - individual components:
  - Testing a single method of an object
  - Testing a single function / procedure
  - Dependencies can be mocked (simulated)
  - Verification of result based on input parameters
  - Verification of side effects through dependency inspection
- Integration testing - integration of components
  - Testing part of a system that uses different components
  - Dependencies are instantiated with testing data
  - Verification of results based on operations on components
- System testing - complete system
  - Testing various use cases
  - Performed on a real system in a testing environment

Apache Maven

Apache Maven is build automation tool used to build and manage Java projects.

Apart from compiling, Maven is used to archive, sign, generate sources and documentation, run tests and deploy or publish the project.

Maven also is a dependency manager. Declared dependencies are automatically downloaded, cached and used during build.

Most IDEs can use Maven for project structure definition allowing for IDE-independent collaboration.

```
my-app
|-- pom.xml
`-- src
      |-- main
      |   |-- java
      |   |   `-- com
      |   |      `-- mycompany
      |   |          `-- app
      |   |               `-- App.java
      |   `-- resources
      |    `-- application.properties
      `-- test
      |-- java
      |   `-- com
      |    `-- mycompany
      |         `-- app
      |              `-- AppTest.java
      `-- resources
           `-- test.properties
```

Unit testing in Java (Maven) project

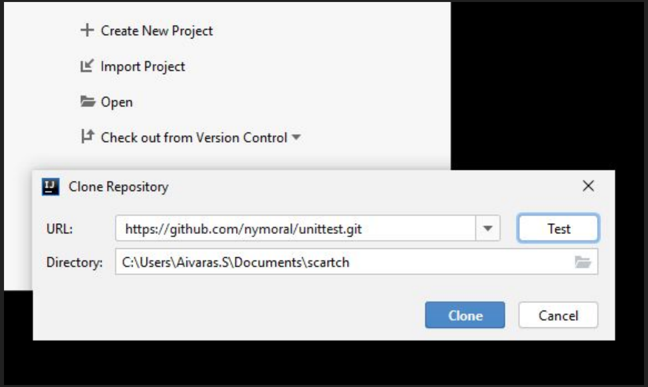
Tests are run by *`maven-surefire-plugin`* plugin.
Place tests in **`src/test/java`** directory.
By convention, use the same package as the class being tested and add **Test* suffix to the class name.
Add `org.junit.jupiter.api.Test` annotation to **each** test method.
Run *mvn test* lifecycle (or *package* / *install*)

```
package lt.mif.unit.exercise1;

import org.junit.jupiter.api.Test;
import java.util.Arrays;
import static org.junit.jupiter.api.Assertions.*;


class StringUtilsTest {
    @Test
    void toCommaSeparatedEmptyList() {
        assertEquals("1, 3", StringUtils.toCommaSeparatedList(Arrays.asList(1, 3)));
    }
}
```

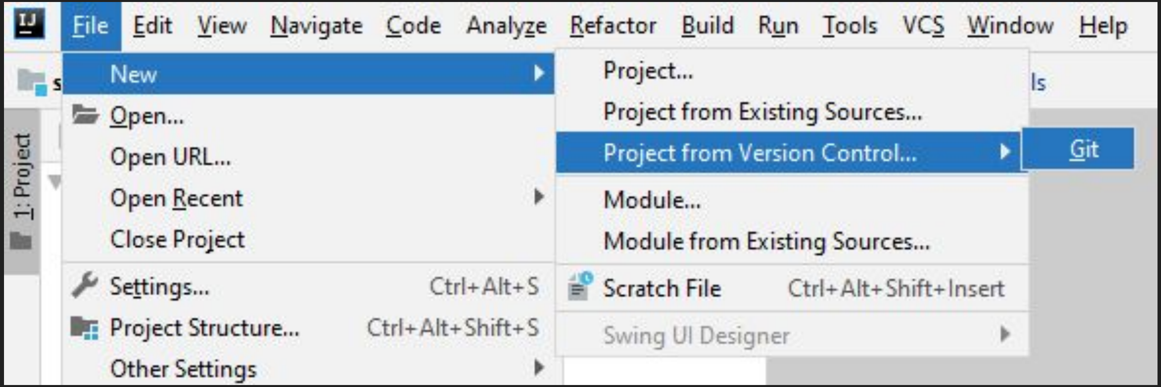
Test driven development

1. Add tests
2. Test fails
3. Write code
4. Test passes
5. Refactor
6. Repeat

Clone a Git project from a Github repository

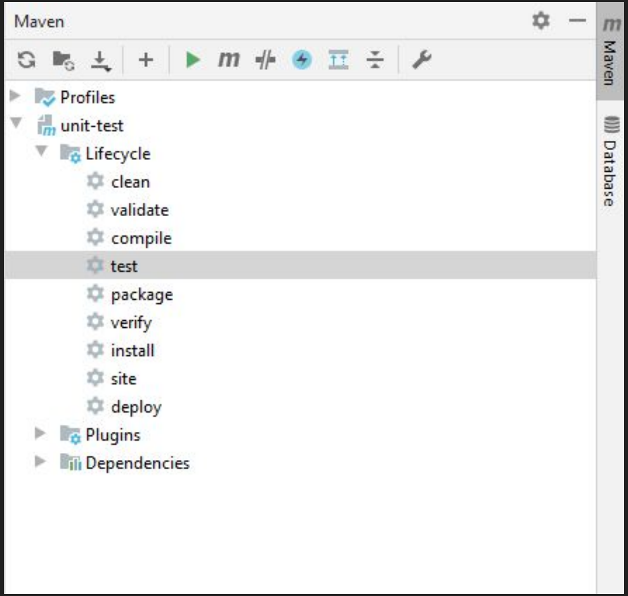


URL: https://github.com/nymoral/unittest.git
Directory: C:\Users\Aivaras.S\Documents\scartch

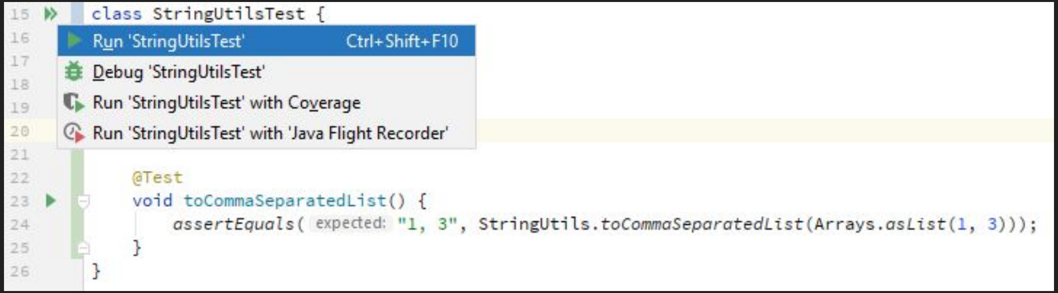


Clone url: https://github.com/nymoral/unittest.git

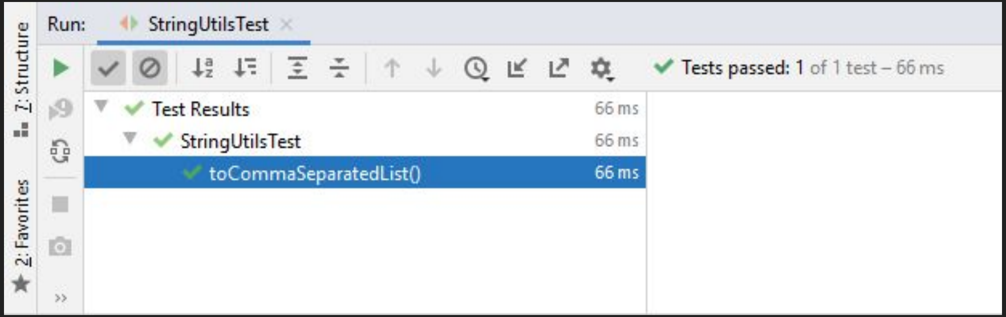
Running tests in Intellij



Maven lifecycle


Run 'StringUtilsTest' Ctrl+Shift+F10
Debug 'StringUtilsTest'
Run 'StringUtilsTest' with Coverage
Run 'StringUtilsTest' with 'Java Flight Recorder'

```
class StringUtilsTest {

    @Test
    void toCommaSeparatedList() {
        assertEquals( expected: "1, 3", StringUtils.toCommaSeparatedList(Arrays.asList(1, 3)));
    }
}
```



```
✓ Tests passed: 1 of 1 test – 66 ms

✓ Test Results                 66 ms
  ✓ StringUtilsTest            66 ms
    ✓ toCommaSeparatedList()   66 ms
```

Intellij test runner with results