WEB TECHNOLOGIES USING JAVA

COURSE 11 UNIT TESTING WITH JUNIT5 AND MOCKITO

AGENDA

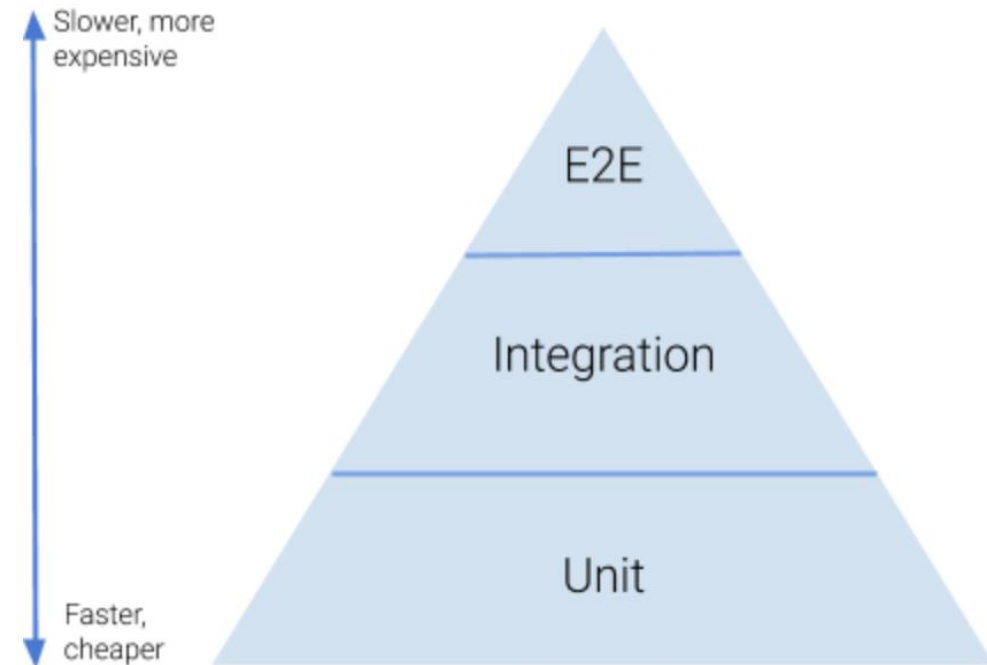- **SOFTWARE TESTING**

- **UNIT TESTING**

- **MOCKING DEPENDENCIES**

SOFTWARE TESTING

- Functional testing:
  - Unit tests: tests that cover one single method
  - Integration tests: tests that cover the system
  - End-to-end tests: tests that cover the system in integration with other systems

- Non-functional testing:
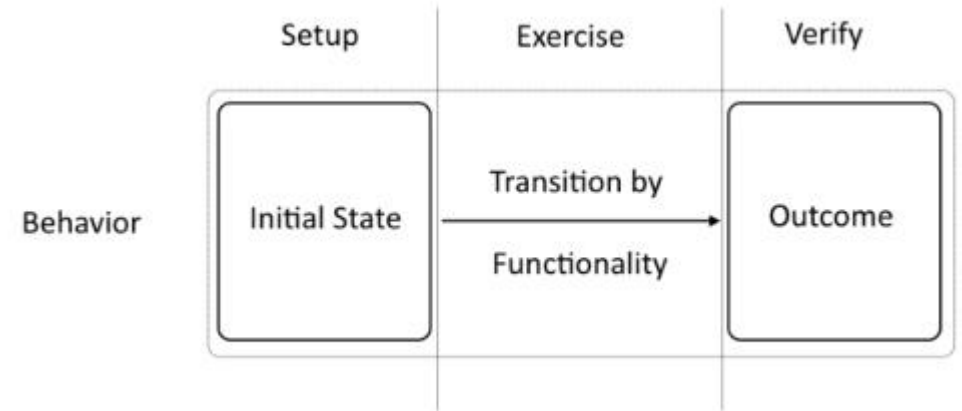 - Performance testing
  - Security testing

UNIT TESTING

- Benefits of unit tests:
  - allow greater test coverage than end-to-end tests
  - increase team productivity
  - detect regressions and limit the need for debugging
  - give us the confidence to refactor and, in general, to make changes
  - improve implementation
  - document expected behavior
  - enable code coverage and other metrics

UNIT TESTING

- Structure of a unit test:
  - setup (arrange, build): this state constitutes well-defined test input and precondition; it is also called the fixture of a test
  - exercise (act, operate): this state is used to invoke the functionality the test intends to check
  - verify (assert, check): this state verifies that the outcome actually matches the specified behavior
  - teardown: this state makes sure the test leaves its environment in the same condition as it was in before the execution of the test.

MOCKING DEPENDENCIES

- mock frameworks:
  - aim to ease stand-in component creation and configuration
  - make it easier to replace third-party components, which otherwise might be expensive to create
  - fluent interface API style
  - configuration is done prior to the exercise phase by means of the tool API. This allows us to stub return values and verify indirect outputs and interactions easily
  - allow for verification of dependent method calls
- well-known mock frameworks: Mockito, PowerMock, EasyMock

BIBLIOGRAPHY

- Java Unit Testing with Junit 5, by Shekhar Gulati, RahulSharma
- Pragmatic Unit testing in Java 8 with Junit, by Jeff Langr

Q&A

THANK YOU

DANIELA SPILCĂ