

Test Driven: From Zero to Hero





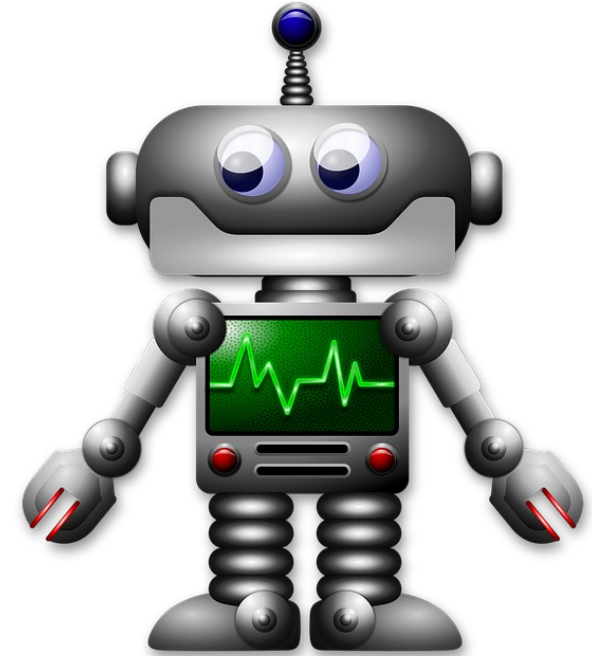
 **ELMAR DOTT**

(M. Schulz) studied at HS Merseburg, Germany, computer science and holds an engineers degree in software engineering. He tweets regularly about several technical topics. The main topics in his field of work are Build and Configuration Management, Software Architecture and Release Management.

About more than 15 years he is working in different large Web Application projects all over the world. He is an independent consultant / trainer. To share his knowledge he gives talks on conferences, if he is not writing on a new article about software engineering. <https://elmar-dott.com>

+ Consultant + Writer + Speaker + Trainer +

- 🚶 Test-Automation?
- 🚶 A fast test automation setup
- 🚶 How to write a unit test
- 🚶 Quality assessment for unit tests
- 🚶 Integration test for Micro Services
- 🚶 Testcontainers for infrastructure



NECESSITIES OF TEST-AUTOMATION

Since the 70' in average 80% of commercial software development projects still failing!

NECESSITIES OF TEST-AUTOMATION

Since the 70' in average 80% of commercial software development projects still failing!

- take too much time → FAIL

NECESSITIES OF TEST-AUTOMATION

Since the 70' in average 80% of commercial software development projects still failing!

- take too much time → FAIL
- the costs increase dramatically → FAIL

NECESSITIES OF TEST-AUTOMATION

Since the 70' in average 80% of commercial software development projects still failing!

- take too much time → FAIL
- the costs increase dramatically → FAIL



Nearly every project I joined the last decade had no single line of automated test!!!

Don't start to write tests for the whole application when you beginning Test Driven Development!



- Open JDK 17 (LTS)
- Apache Maven 3.8
- JUnit 5
- Docker



MAVEN SUREFIRE PLUGIN

```
<build>  
  <plugins>  
    <plugin>  
      <groupId>org.apache.maven.plugins</groupId>  
      <artifactId>maven-surefire-plugin</artifactId>  
      <version>3.0.0</version>  
    </plugin>  
  </plugins>  
</build>
```

CONFIGURATION

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>3.0.0</version>
      <configuration>
        <testFailureIgnore>true</testFailureIgnore>
      </configuration>
    </plugin>
  </plugins>
</build>
```

WRITING TESTABLE CLEAN CODE

```
public boolean validate(String content, String regex) {
    boolean test = false;

    return test;
}
```

WRITING TESTABLE CLEAN CODE

```
public boolean validate(String content, String regEx) {  
    boolean test = false;  
  
    if (content.matches(regEx)) {  
        test = true;  
  
    } else {  
  
        String msg = "validate('" + regEx + "') did not match "  
                    + content;  
        LOGGER.log(msg, LogLevel.WARN);  
    }  
  
    return test;  
}
```

READABLE TEST CASES

```
public class ValidatorTest {  
  
    @Test  
    void validate() {  
        boolean check = Validator.validate('abc', 'abc');  
        assertTrue(check);  
    }  
  
}
```

FALSE POSITIVE

```
public class ValidatorTest {  
  
    @Test  
    void validate() {  
        Boolean check = Validator.validate('abc', 'abc');  
        assertTrue(check);  
    }  
  
    @Test  
    void failValidate() {  
        Boolean check = Validator.validate('ABC', 'abc');  
        assertFalse(check);  
    }  
}
```

THE AAA PRINCIPLE

Function:

```
public T functionToTest(...) {  
    return true;  
}
```

AAA-Principle

- ✓ **Arrange** (setup for test)
- ✓ **Act** (execute function)
- ✓ **Assert** (compare result)

Test:

```
public void doSomethingTest() {  
    //Arrange  
    T result;  
    T expected = ...;  
    Object o = new Object();  
    //Act  
    result = o.functionToTest(...);  
    //Assert  
    assert.equals(expected, result);  
}
```


How many test cases will be needed?

MCCABE COMPLEXITY METRIC

Algorithm:

counting loops and conditions and add 1.

→ if: expression-1 + expression-2 = 2
add 1
Complexity = 3

3 test cases will be needed

MCCABE COMPLEXITY METRIC

```
public boolean isEmpty(String message) {  
    boolean test = false;  
    if (message != null || message.equals("")) {  
        test = true;  
    } else {  
        LOGGER.log("validate('" + regEx + "') did not match "  
            + content, LogLevel.WARN);  
    }  
    return test;  
}
```

BRANCH VS LINE COVERAGE

```
207.     @Override
208.     public void addNode(final TreeNode node) {
209.         boolean add = false;
210.         if (!this.tree.isEmpty()
211.             && node != null
212.             && !node.getUuid().equals(this.rootUuid)) {
213.
214.             for (TreeNode check : this.tree) {
215.
216.                 if (!node.getUuid().equals(check.getUuid())
217.                     || !node.getParent().equals(check.getParent())
218.                     && !node.getNodeName().equals(check.getNodeName())) {
219.
220.                     add = true;
221.                     LOGGER.log("Node [" + node.getNodeName() + "] added.",
222.                               LogLevel.DEBUG);
223.                     break;
224.
225.                 } else {
226.                     LOGGER.log("Node with the same name and parent or the same UUID already exist.",
227.                               LogLevel.WARN);
228.                 }
229.             }
230.
231.             if (add) {
232.                 tree.add(node);
233.             }
234.         }
235.     }
```

MAVEN FAILSAFE PLUGIN

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-failsafe-plugin</artifactId>
  <version>3.0.0</version>
  <executions>
    <execution>
      <phase>integration-test</phase>
      <goals>
        <goal>integration-test</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

MAVEN FAILSAFE PLUGIN

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-failsafe-plugin</artifactId>
  <version>3.0.0</version>
  <executions>
    <execution>
      <phase>integration-test</phase>
      <goals>
        <goal>integration-test</goal>
      </goals>
    </execution>
  </executions>

  <configuration>
    <skipTests>${skipIntegrationTests}</skipTests>
    <testFailureIgnore>true</testFailureIgnore>
  </configuration>
</plugin>
```

MICRO SERVICE

```
@GET
@Path("/{account}")
@Produces({MediaType.APPLICATION_JSON})
public Response fetchAccount(@PathParam("account") String accountId) {
    Response response = null;
    try {
        AccountDO account = accountDAO.find(accountId);
        if (account != null) {
            String json = accountDAO.serializeAsJson(account);
            response = Response.status(Response.Status.OK)
                .type(MediaType.APPLICATION_JSON)
                .entity(json)
                .encoding("UTF-8")
                .build();
        } else {
            response = Response.status(Response.Status.NOT_FOUND).build();
        }
    } catch (Exception ex) {
        response = Response.status(Response.Status.INTERNAL_SERVER_ERROR).build();
    }
    return response;
}
```

Testing RESTful Services



Project
Grizzly

```
public class EmbeddedGrizzly {
    public static final String BASE_URI = "http://localhost:8888/";
    public static final String BASE_PACKAGE = "org.europa.together";

    public static HttpServer startServer() {

        HttpServer server = null;
        try {
            final ResourceConfig rc = new ResourceConfig().packages(BASE_PACKAGE);

            server = GrizzlyHttpServerFactory.createHttpServer(URI.create(BASE_URI), rc);
            server.start();
            System.in.read();

        } catch (Exception ex) {
            System.err.println(ex.getMessage());
        }
        return server;
    }
}
```

<https://javaee.github.io/grizzly/>

INTEGRATION TEST

```
public class AccountServiceIT {  
    private static HttpServer server;  
    private static WebTarget target;  
  
    @BeforeAll  
    static void setUp() {  
        try {  
            server = EmbeddedGrizzly.startServer();  
            ClientConfig config = new ClientConfig();  
            Client client = ClientBuilder.newClient(config);  
            target = client.target(EmbeddedGrizzly.BASE_URI);  
        } catch (Exception ex) {  
            LOGGER.catchException(ex);  
        }  
        Assumptions.assumeTrue(server.isStarted(), "REST Server startup failed.");  
    }  
  
    @AfterAll  
    static void tearDown() {  
        server.shutdownNow();  
    }  
}
```

TEST CASE

```
@Test
void getAccountStatus404() {
    LOGGER.log("TEST CASE: getAccount() 404 : NOT FOUND", LogLevel.DEBUG);

    Response response = target
        .path(API_PATH).path("/NotExist")
        .request()
        .accept(MediaType.APPLICATION_JSON)
        .get(Response.class);

    assertEquals(404, response.getStatus());
}
```

- Setting up an automated infrastructure like database server
- Needs an docker installation.
- Is included in the test cases
- Can extended by own Testcontainers
- Should configured as Integration Test

PREPARING DEPENDENCIES

```
<dependencies>
  <!-- testcontainer -->
  <dependency>
    <groupId>org.testcontainers</groupId>
    <artifactId>postgresql</artifactId>
    <version>1.17.6</version>
    <scope>test</scope>
  </dependency>

  <!-- database driver -->
  <dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.5.1</version>
  </dependency>
</dependencies>
```

<https://www.testcontainers.org>

TEST CASE

```
public class DatabaseIT {  
    @ClassRule  
    public static PostgreSQLContainer postgresSQLContainer  
        = new PostgreSQLContainer("postgres:14")  
            .withDatabaseName("tests-db")  
            .withUsername("postgres")  
            .withPassword("postgres");  
  
    @Test  
    void connectionTest() {  
        JdbcConnection jdbc = new JdbcConnection();  
        jdbc.connect("jdbc:tc:postgresql:14:///test-db");  
        try {  
            jdbc.executeQuery("CREATE TABLE IF NOT EXISTS test (column_1 int, column_2 char(255));");  
            jdbc.executeQuery("INSERT INTO test (column_1, column_2) VALUES (11, 'test entry');");  
        } catch (SQLException ex) {  
            System.err.print(ex.getMessage());  
        }  
    }  
}
```

- Just write test cases for your implementation you currently working on.

- Just write test cases for your implementation you currently working on.
- Keep your test methods compact.

- Just write test cases for your implementation you currently working on.
- Keep your test methods compact.
- Use Logging to detect problems faster.

- Just write test cases for your implementation you currently working on.
- Keep your test methods compact.
- Use Logging to detect problems faster.
- Try out Testcontainers for infrastructure installations.



Homepage : <https://elmar-dott.com>
GitHub : <https://github.com/ElmarDott>
AnchorFM : <https://anchor.fm/elmar-dott>
Twitter : <https://twitter.com/ElmarDott>
Speaker Deck : <https://speakerdeck.com/elmardott>

Lbry : <https://lbry.tv/@elmar.dott:8>
BitChute : <https://www.bitchute.com/channel/3IyCzKdX8Ip0/>



Danke / Спасибо / Gracias

REFERENCES

[1] Marco Schulz, 2021, Continuous Integration mit Jenkins, Rheinwerk, ISBN: ISBN 978-3-8362-7834-8
<https://www.rheinwerk-verlag.de/continuous-integration-mit-jenkins/>

[2] Homepage: <https://elmar-dott.com>

[3] GitHub: <https://github.com/ElmarDott>

[4] UnitTesting Antipattern:

<https://www.yegor256.com/2018/12/11/unit-testing-anti-patterns.html>

[5] TDD Misbeliefs:

<https://www.javacodegeeks.com/2019/07/tdd-misbeliefs.html>

[6] When TDD doesn't work:

<https://blog.cleancoder.com/uncle-bob/2014/04/30/When-tdd-does-not-work.html>





BITCHUTE

