

Testramverk - Evosuite, JUnit 5 (Mockito, PIT)

Fredrik & Stefan

Länk till presentationen

<https://goo.gl/8T3h4f>

Evosuite

- Evosuite kan generera tester som täcker de gränsvärden du har i dina ifsatser
- I teorin intressant
- I praktiken buggigt :(
- Funkar hjälpligt på user-manager på mac, men inte på current-race
- Funkar uselt på Linux (Ubuntu 14.04 och 16.04)
- Vet ej om det är för att vi kör spring boot eller om deras runtimelösning för att exekvera testerna suger
- Kanske intressant om de får det att fungera bättre.

Ladda ner och installera

- <http://www.evosuite.org/>
- Stoppa in deras repo i pom (finns ej i maven central)
- Sätt upp plugin för build steget så att testerna kan genereras
- `mvn -DmemoryInMB=2000 -Dcores=4 evosuite:generate evosuite:export test`

Preparerad POM finns i user-manager i mitt repo

<https://github.com/ogr3/race-management-system>

i branchen evosuite

`git checkout -b evosuite origin/evosuite`

POM

```
<profile>
  <id>evosuite</id>
  <activation>
    <activeByDefault>false</activeByDefault>
  </activation>
  <properties>
    <evosuite-version>1.0.3</evosuite-version>
  </properties>
  <dependencies>
    <!-- Evosuite test generator -->
    <dependency>
      <groupId>org.evosuite</groupId>
      <artifactId>evosuite-standalone-runtime</artifactId>
      <version>${evosuite-version}</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  ...
</profile>
```

<build>

```
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>2.19.1</version>
    <configuration>
      <shutdown>exit</shutdown>
      <properties>
        <property>
          <name>listener</name>
          <value>org.evosuite.runtime.InitializingListener</value>
        </property>
      </properties>
    </configuration>
  </plugin>
</plugins>
<pluginManagement>
  <plugins>
```

POM repos

```
<pluginRepositories>
  <pluginRepository>
    <id>EvoSuite</id>
    <name>EvoSuite Repository</name>
    <url>http://www.evosuite.org/m2</url>
  </pluginRepository>
</pluginRepositories>
```

```
<repositories>
  <repository>
    <id>EvoSuite</id>
    <name>EvoSuite Repository</name>
    <url>http://www.evosuite.org/m2</url>
  </repository>
</repositories>
```

Vad hände i LiiV?

Innan:

Tests run: 672, Failures: 0, Errors: 0, Skipped: 0

Efter:

Tests run: 1985, Failures: 2, Errors: 33, Skipped: 0

Så typ 1300 genererade testfall, men man måste gå igenom för att se de som failar.

JUnit5 - Next generation

- Består nu av 3 olika subprojekt/moduler
 - JUnit Jupiter (Nya testmotorn för JUnit 5)
 - JUnit Platform (Grundstommen för att köra tester (JUnit 4 & JUnit 5))
 - JUnit Vintage (Testmotor för JUnit 4)
- Java 8
- Inbyggt i IntelliJ (ännu lite buggigt)
- Eclipse - ännu inget inbyggt stöd

JUnit 5 och spring boot

- Lite tidigt att införa än - kräver en plugin som inte finns mer än på github
- <https://github.com/sbrannen/spring-test-junit5>

```
@SpringBootTest
@ExtendWith(SpringExtension::class)
class DataBeanTest : ApplicationTest() {

    @Autowired lateinit var dataBean: DataBean

    @Test
    fun testAutowiring() {
        assertNotNull(dataBean)
    }
}
```

JUnit5 - @DisplayName

```
@DisplayName("A special test case")
class DisplayNameDemo {

    @Test
    @DisplayName("Custom test name containing spaces")
    void testWithDisplayNameContainingSpaces() {

    }

    @Test
    @DisplayName("J°□°) J")
    void testWithDisplayNameContainingSpecialCharacters() {

    }

    @Test
    @DisplayName("🤖")
    void testWithDisplayNameContainingEmoji() {

    }

}
```

JUnit5 - Assertions

```
@Test
public void lambdaExpressions() {
    // lambda expression for condition
    assertTrue(() -> "".isEmpty(), "string should be empty");

    // lambda expression for assertion message
    assertEquals("foo", "foo", () -> "message is lazily evaluated");
}
```

JUnit5 - Assertions

```
@Test
void groupedAssertions() {
    // In a grouped assertion all assertions are executed, and any
    // failures will be reported together.
    assertAll("address",
        () -> assertEquals("John", address.getFirstName()),
        () -> assertEquals("User", address.getLastName())
    );
}

@Test
void exceptionTesting() {
    Throwable exception = expectThrows(IllegalArgumentException.class, () -> {
        withdrawTooMuchMoney(100 000 kr);
    });
    assertEquals("Not enough money!", exception.getMessage());
}
```

JUnit5 - Migrering från Junit 4

- Inga fler Runners → `@ExtendWith(MockitoExtension.class)`
- Båda versionerna kan samexistera

JUnit5 - Labbar

- <https://github.com/junit-team/junit5>
- <http://junit.org/junit5/docs/current/user-guide/>
- Ladda ned och labba själv:
<https://github.com/nilden/CAG-JUnit5-Lab.git>

Mockito

- spy
- ArgumentCaptor
- hur mocka servletContext och andra grejer för JSF - exempel
- In memory mongo

JSF special - FacesContext

För att klara av att simulera en runtimemiljö där det mesta kretsar kring facesContext får man skapa en lite mer avancerad mock.

Görs med egen klass och ett par rader i @Before och @After testmetoderna

Mocka FacesContext

```
public abstract class ContextMocker extends FacesContext {
    private ContextMocker() {

    }

    private static final Release RELEASE = new Release();

    private static class Release implements Answer<Void> {
        @Override
        public Void answer(InvocationOnMock invocation) throws Throwable {
            setCurrentInstance(null);
            return null;
        }
    }
    ...
}
```

Mocka FacesContext fortsättning

...

```
public static FacesContext mockFacesContext() {  
    FacesContext context = Mockito.mock(FacesContext.class);  
    setCurrentInstance(context);  
    Mockito.doAnswer(RELEASE)  
        .when(context)  
        .release();  
    return context;  
}
```

Användning

```
@Mock
private FacesContext facesContext;

@Mock
private UIViewRoot viewRoot;

@Before
public void setup() {
    MockitoAnnotations.initMocks(this);
    facesContext = ContextMocker.mockFacesContext();
    when(facesContext.getViewRoot()).thenReturn(viewRoot);
    when(viewRoot.getLocale()).thenReturn(Locale.forLanguageTag("sv"));
}

@After
public void teardown() {
    facesContext.release();
}
```

ArgumentCaptor

Oftast använder man matchers för att verifiera att interaktionen med en mock har gjorts med vissa argument.

Om argumentet är lite mer komplext som ett större objekt där man vill validera specifika fält spå kan man fånga det anropade argumentet med en `ArgumentCaptor` och sedan kika på hur det såg ut.

Om argumentet används flera gånger i flera metodanrop får man en lista med fångade argument i den ordningen de anropades

ArgumentCaptor exempel

```
ArgumentCaptor<FacesMessage> facesMessageArgumentCaptor = ArgumentCaptor.forClass(FacesMessage.class);

verify(facesContext, atLeastOnce()).addMessage(eq("searchForm:sokText"), facesMessageArgumentCaptor.capture());

FacesMessage facesMessage = facesMessageArgumentCaptor.getValue();
assertEquals("text att jämföra", facesMessage.getSummary());
```

Använda spy för att fånga new Object

Om koden implementerar new Object() så kan det ju vara svårt att fixa mockar

```
private ServiceLocatorWS serviceLocatorWS;

private LiivForetag liivForetag;

@Before
public void setup() {
    serviceLocatorWS = Mockito.spy(new ServiceLocatorWS());
    liivForetag = Mockito.mock(LiivForetag.class);
    doReturn(liivForetag).when(serviceLocatorWS).getService();
}
```

Nu kan man interagera med liivForetag som vilken mock som helst

RestAssured

SensorControllerTest

given()

.body(bodyAsString).with().contentType(ContentType.JSON)

.when()

.get(SensorController.REGISTER_SENSOR_URL)

.then()

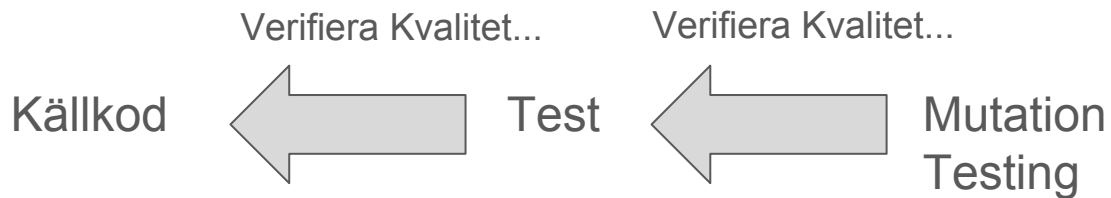
.statusCode(HttpStatus.METHOD_NOT_ALLOWED.value());

PIT - muterande tester

<https://www.youtube.com/watch?v=S8po8Gg6IJY>

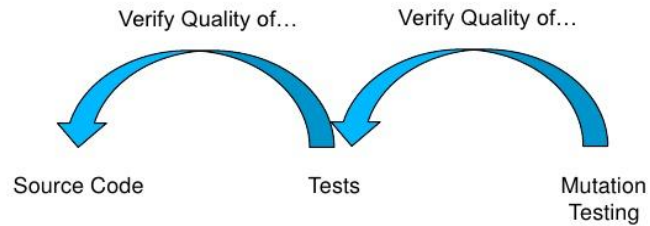
PIT - Mutation Testing

Ramverk för att testa kvalitet i tester



PIT - Mutation Testing

Ramverk för att testa kvalitet på tester



PIT - Mutation Testing

- Traditionell testtäckning
 - mäter endast vilken kod som körs genom dina tester
 - identifiera vad som inte testas
 - mäter inte att testet testar rätt saker

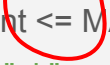
PIT - Mutation Testing

- PIT introducerar små förändringar i koden (mutationer) och kör dina enhetstester mot dessa förändringar.
- När koden förändras bör enhetstesterna 'faila'.
- Om inte enhetstesterna 'failar' kan det vara en indikation på att det finns ett problem med testet

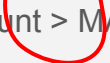
PIT - Mutation Testing

En mutation är en (liten) förändring i koden

```
if ( amount <= MAX_AMUONT ) {  
    return "ok";  
} else {  
    return "not ok";  
}
```



```
if ( amount > MAX_AMUONT ) {  
    return "ok";  
} else {  
    return "not ok";  
}
```



PIT - Mutation Testing

Aktiva mutators per default

- Conditionals Boundary Mutator
- Increments Mutator
- Invert Negatives Mutator
- Math Mutator
- Negate Conditionals Mutator
- Return Values Mutator
- Void Method Calls Mutator

PIT - Mutation Testing

Inaktiva mutators per default

- Constructor Calls Mutator
- Inline Constant Mutator
- Non Void Method Calls Mutator
- Remove Conditionals Mutator
- Experimental Member Variable Mutator
- Experimental Switch Mutator

PIT - Labbar

- Ladda och lägg till plugin i pom.xml

```
<plugin>
  <groupId>org.pitest</groupId>
  <artifactId>pitest-maven</artifactId>
  <version>LATEST</version>
  <configuration>
    <targetClasses>

<param>com.your.package.root.want.to.mutate*</param>
    </targetClasses>
    <targetTests>
      <param>com.your.package.root*</param>
    </targetTests>
  </configuration>
</plugin>
```

IDE Plugins:

- <http://plugins.jetbrains.com/plugin/?idea&pluginId=7119>
- <https://github.com/philglover/pitclipse>

Labb:

<https://github.com/nilden/CAG-PITTest-lab.git>

Generera PIT rapport

```
mvn -Ppitest -DwithHistory org.pitest:pitest-maven:mutationCoverage
```

```
<profiles>
  <profile>
    <id>pitest</id>
    <activation>
      <activeByDefault>false</activeByDefault>
    </activation>
    <build>
      <plugins>
        ...
      </plugins>
    </build>
  </profile>
```

<plugin>

```
<groupId>org.pitest</groupId>
<artifactId>pitest-maven</artifactId>
<version>1.1.10</version>
<configuration>
  <targetClasses>
    <param>se.ehalsomyndigheten.liiv.web.*</param>
  </targetClasses>
  <targetTests>
    <param>se.ehalsomyndigheten.liiv.web.*</param>
  </targetTests>
  <mutators>
    <mutator>CONDITIONALS_BOUNDARY</mutator>
    <mutator>INCREMENTS</mutator>
    <mutator>INVERT_NEGS</mutator>
    <mutator>MATH</mutator>
    <mutator>NEGATE_CONDITIONALS</mutator>
    <mutator>RETURN_VALS</mutator>
    <mutator>VOID_METHOD_CALLS</mutator>
    <mutator>CONSTRUCTOR_CALLS</mutator>
    <mutator>INLINE_CONSTS</mutator>
    <mutator>NON_VOID_METHOD_CALLS</mutator>
    <mutator>REMOVE_CONDITIONALS</mutator>
```