Зборник радова Факултета техничких наука, Нови Сад

УДК: (Upisuje redakcija, ostaviti ovaj red)
ДОИ: (Upisuje redakcija, ostaviti ovaj red)

КОМПАРАТИВНА АНАЛИЗА *MAVEN SUREFIRE* И *ALLURE* РАДНИХ ОКВИРА ЗА ГЕНЕРИСАЊЕ ТЕСТ ИЗВЕШТАЈА

COMPARATIVE ANALYSIS OF MAVEN SUREFIRE AND ALLURE TEST REPORT FRAMEWORKS

Ана Гавриловић, *Факултет техничких наука, Нови Сад*

Област –ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *Овај рад представља компаративну анализу радних оквира за генерисање извештаја о тестовима Maven Surefire и Allure, користећи као студију случаја веб апликацију за биоскоп и GitHub Actions pipeline-ове. Рад евалуира метрике као што су време генерисања извештаја, употреба ресурса и величина извештаја, као и кориснички интерфејс, опције прилагођавања и лакоћа интеграције у оквиру CI/CD pipeline-а, истичући предности и мане сваког оквира. Резултати су показали да се Maven Surefire истиче брзином и ефикасношћу, генеришући мање извештаје са мањом потрошњом ресурса, док Allure пружа детаљније, прилагодљиве извештаје, али уз већу цену у погледу перформанси и коришћења ресурса.*

Кључне речи: *тестирање софтвера, тест извештаји, Allure Reports, Maven Surefire Report*

Abstract – *This thesis presents a comparative analysis of the Maven Surefire and Allure test report frameworks, using the cinema web application and GitHub Actions pipelines as a case study. The paper evaluates metrics such as report generation time, resource usage, and report size, as well as user interface, customization options, and ease of integration within a CI/CD pipeline, highlighting the strengths and weaknesses of each framework. The results showed that Maven Surefire excels in speed and efficiency, generating smaller reports with less resource consumption, while Allure provides more detailed, customizable reports, but at a higher cost in terms of performance and resource usage.*

Keywords: *software testing, test reports, Allure Reports, Maven Surefire Report*

1. УВОД

У области развоја софтвера, тестирање и извештавање играју важну улогу у одржавању квалитета кода, а са све већом сложеношћу софтверских система, потреба за ефикасним алатима за извештавање је значајно порасла. Избор одговарајућег оквира за извештавање је изазован због разноврсности доступних опција и фактора као што су перформансе, лакоћа интеграције, јасноћа извештаја, скалабилност и употребљивост. Овај рад има за циљ да кроз компаративну анализу *Maven Surefire* и *Allure* радних оквира процени њихове предности и мане у контексту поменутих фактора, пружајући увид у то који је алат боље прилагођен специфичним потребама тестирања.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Бранко Милосављевић, ред. проф.

2. СПЕЦИФИКАЦИЈА СИСТЕМА

Ефикасност упоредних анализа у великој мери се ослања на добро дефинисан и структуриран систем. Ово поглавље пружа детаљан преглед система који се користи као студија случаја за ово истраживање.

2.1. Веб апликација за биоскоп

Радни оквири за које је вршена компаративна анализа генеришу тест извештаје за тестове *full-stack* веб апликације намењене за употребу биоскопа. Примарна сврха апликације је да олакша онлајн резервацију карата и управљање, за раднике и посетиоце биоскопа. Неке од основних функционалности апликације су преглед филмова, преглед пројекција, резервације пројекција, преглед биоскопских сала, кориснички налози, админ панел и друге.

Frontend је развијен коришћењем *React*-а, *backend* је имплементиран у програмском језику Java и радном оквиру *Spring Boot*, док за базу података апликација користи *PostgreSQL*.

2.2. Тестирање софтвера

За биоскопску веб апликацију, усвојен је вишеструки приступ тестирању, који укључује више врста тестова који покривају различите аспекте апликације. Коришћена је комбинација *unit* тестова, тестова интеграције и E2E (*end-to-end*) тестова. *Unit* тестови представљају основну јединицу тестирања јер се фокусирају на појединачне компоненте у изолованом окружењу. Они осигуравају да свака јединица кода ради како је очекивано. За писање *unit* тестова коришћени су *JUnit* и *Mockito*. Интеграциони тестови служе за тестирање интеракција између различитих компоненти апликације, на пример *backend*-а и базее података. E2E тестови обезбеђују процену рада целе апликације, симулирајући стварне корисничке сценарије како би се потврдило да систем ради како је предвиђено од почетка до краја. Ови тестови су написани помоћу *Selenium* алата.

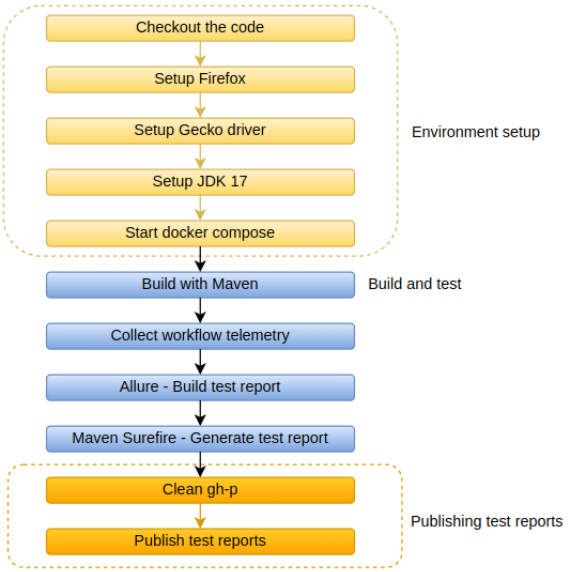
Како би се додатно повећала покривеност тестовима за најразличитије вредности улазних параметара, већина тестова је параметризована. Параметризовани тестови представљају напредну технику тестирања, где се иста тестна логика покреће са различитим улазним параметрима. На овај начин, један тестни случај може покрити широк спектар сценарија и граничних случајева. У контексту упоредне анализе алата за генерисање тест извештаја, параметризовани тестови су били посебно значајни јер се помоћу њих може симулирати веома велики број тест случајева, какав се очекује код већине реалних апликација.

Укупан број тест случајева је 41143, од чега су 31000 *unit* тестови, 10019 интеграциони и 123 E2E тестови.

2.3. Дизајн CI pipeline-а

Токови континуиране интеграције (енгл. *Continuous Integration pipelines* - CI *pipelines*) играју кључну улогу у савременим праксама развоја софтвера, јер аутоматизују процес интегрисања кода написаног од стране више програмера у један заједнички репозиторијум, тако што извршавају све акције које су неопходне пре саме интеграције.

За ову апликацију, написан је CI *pipeline* који је дизајниран тако да аутоматизује процесе *build*-а, тестирања и генерисања извештаја тестирања. У два одвојена корака *pipeline*-а, генеришу се два тест извештаја, један од стране *Maven Surefire* алата, а други од стране *Allure* радног оквира. Пре ових корака, врши се неколико других корака који служе за подешавање окружења за даљи ток *pipeline*-а. Такође је подешена и акција која мери перформансе и потрошњу ресурса корака који генеришу тест извештаје. На слици 1 приказани су кораци CI *pipeline*-а биоскопске апликације.



Слика 1. *Ток и кораци CI pipeline-а апликације*

2.4. Maven Surefire Report

У оквиру алата за аутоматизацију *Apache Maven*-а, постоји више додатка (енгл. *pulgin*) који доприносе процесу тестирања. Најзначајнији додатак је *Maven Surefire Plugin*, дизајниран посебно за покретање тестова. Овај додатак генерише извештаје о тестирању у формату обичног текста или XML формату. Међутим, у овим форматима, тест извештаји нису читљиви обичном кориснику, те их је пожељно конвертовати у HTML формат.

За те сврхе користи се *Maven Surefire Report Plugin [1]*, чија је сврха да парсира тест артефакте из XML формата у HTML фајл, који се може отворити и прегледати у веб претраживачу. Ти извештаји пружају детаљне информације о извршењу теста, укључујући број покренутих, успешних, неуспешних и прескочених тестова, дистрибуцију тестова по пакетима, поруке о грешкама и друге.

За интеграцију оба *pugin*-а потребно их је додати у конфигурацију *Maven* пројекта, затим покренути тестове, и након тога помоћу команде *surefire-report:report-only* изгенерисати извештај.

2.5. Allure Reports

Allure [2] оквир за извештаје је алат дизајниран за генерисање детаљних и визуелно привлачних извештаја о тестирању. Има богат скуп функција и прилагодљив је различитим потребама тестирања. *Allure* подржава широк спектар тест оквира и језика. Овај радни оквир користи тест артефакте добијене након извршавања тестова од стране коришћеног радног оквира за те сврхе и, на основу њих, генерише тест извештаје. Добијени тест извештај представља статичку веб страницу са приказом резултата извршавања тестова, њиховом броју, подељености по категоријама, пакетима, и многобројним графиконима.

Allure се састоји од адаптера за радни оквир и *allure* програма за командну линију. Тест извештаји се генеришу кроз следећа три корака:

1. Инсталирање *Allure* адаптера и програма за командну линију, и додавање подршке за поменути адаптер у оквиру конфигурације пројекта
2. Покретање тестова на исти начин на који би се покретали у технологијама у којима је пројекат имплементиран
3. Генерисање HTML извештаја користећи једну од команди из програма командне линије (*allure generate* или *allure serve*) [3]

Allure има подршку за интеграцију са различитим алатима, као што су *Azure DevOps, GitHub Actions, Jenkins, JetBrains IDEs, Visual Studio Code* и другим. У том случају потребно је додати одговарајућу конфигурацију у односу на алат у који се *Allure* интегрише.

3. ИМПЛЕМЕНТАЦИЈА СИСТЕМА

Ово поглавље има за циљ да пружи увид у практичну имплементацију и техничке детаље делова система који су неопходни како би се извршила упоредна анализа радних оквира за генерисање тест извештаја.

3.1. Конфигурација CI pipeline-а

CI *pipeline* за апликацију биоскопа је имплементиран помоћу *GitHub Actions* алата. *GitHub Actions [4]* представља CI/CD платформу која је интегрисана у оквиру *GitHub* екосистема. *GitHub Actions* омогућава креирање прилагодљивих токова посла (енгл. *workflows*) писањем YAML конфигурационих фајлова.

У главном (енгл. *root*) директоријуму *GitHub* репозиторијума, креиран је нови директоријум *.github/workflows* и у њему документ *ci.yml*. Почетни део документа, у ком се специфицирају назив *pipeline*-а и догађаји га окидају, као и кораци који су задужени за подешавање окружења и за извршавање *build*-а дати су на листнгу 1.

```yaml
name:  CI pipeline for Cinema Application

on:
  pull_request:
    branches: [ "master" ]
  push:
    branches: [ "*" ]

jobs:
  build-and-test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout the code
        uses: actions/checkout@v4.1.1

      - name: Setup Firefox
        uses: browser-actions/setup-firefox@latest

      - name: Setup gecko driver
        uses: browser-actions/setup-geckodriver@latest

      - name: Set up JDK 17
        uses: actions/setup-java@v3
        with:
          java-version: '17'
          distribution: 'temurin'
          cache: maven

      - name: Start docker compose
        run: docker-compose up -d

      - name: Build with Maven
        run: mvn -B package -DGECKODRIVER_PATH=$(which geckodriver) \
                    -DFIREFOX_PATH= --file backend/pom.xml
```

Листнг 1. *CI pipeline - почетна конфигурација*

3.2. Конфигурација Github акције за мерење перформанси

За прикупљање телеметријских података у оквиру CI *pipeline*-а, коришћена је *catchpoint/ workflow-telemetry-action* [5] *GitHub* акција, верзије 1.8.7. Ова акција је дизајнирана тако да хвата метрике перформанси током извршавања CI *pipeline*-а. Интеграцијом ове акције можемо пратити различите аспекте *pipeline*-а, као што су коришћење процесора, потрошња меморије, диска, активност мреже, као и времена извршавања различитих корака. На листингу 2 приказана је конфигурација ове акције у оквиру *pipeline*-а.

```yaml
- name: Collect Workflow Telemetry
  uses: catchpoint/workflow-telemetry-action@v1.8.7
  with:
    metric_frequency: 1s
```

Листинг 2. *Конфигурација акције за мерење перформанси корака pipeline-а*

3.3. Конфигурација и генерисање Maven Surefire извештаја

За *build* и *packaging Spring Boot* апликације користи се *spring-boot-maven-plugin*, док *maven-surefire-report-plugin* генерише извештаје о тестирању. На листингу 3 представљен је начин на који се ови *plugin*-ови додају у пројекат, у оквиру *pom.xml* фајла, а на листингу 4 дата је конфигурација корака *pipeline*-а у оквиру којег се генеришу *Maven Surefire* извештаји.

```xml
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>

<!-- Maven surefire test reports plugin -->
<reporting>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-report-plugin</artifactId>
            <version>3.2.2</version>
        </plugin>
    </plugins>
    <outputDirectory>github-pages/surefire</outputDirectory>
</reporting>
```

Листинг 3. *Plugin-ови за Maven Surefire у оквиру pom.xml*

```yaml
- name: Maven Surefire - Generate test report
  run: |
    cd ./backend
    mvn surefire-report:report-only -DreportOutputDirectory=github-pages/surefire/
```

Листинг 4. *Конфигурација корака за генерисање Maven Surefire тест извештаја*

3.4. Конфигурација и генерисање Allure извештаја

За интеграцију *Allure* радног оквира у пројекат, прво је потребно укључити *allure-junit5 dependency* у пројекат (листинг 5). Овај *dependency* осигурава да *Allure* може да обради постојеће резултате тестова и генерише извештаје на основу њих.

```xml
<dependency>
    <groupId>io.qameta.allure</groupId>
    <artifactId>allure-junit5</artifactId>
    <scope>test</scope>
    <version>2.13.0</version>
</dependency>
```

Листинг 5. *Dependency за Allure Reports у pom.xml*

Након додавања зависности, следећи корак је конфигурација CI *pipeline*-а да покрене *Allure* акцију за генерисање извештаја и то је приказано на листингу 6.

```yaml
- name: Allure - Load test report history
  uses: actions/checkout@v4.1.1
  if: always()
  continue-on-error: true
  with:
    ref: gh-p
    path: gh-p

- name: Allure - Build test report
  uses: simple-elf/allure-report-action@v1.9
  if: always()
  with:
    allure_history: ./backend/github-pages
    allure_results: ./backend/target/allure-results
    subfolder: allure-history
```

Листинг 6. *Конфигурација корака за генерисање Allure тест извештаја*

4. РЕЗУЛТАТИ

У овом поглављу су дате вредности метрика као и квалитативних особина радних оквира за генерисање извештаја о тестовима *Maven Surefire* и *Allure*. CI *pipeline* је извршен више пута и иако метрике благо варирају, сваки пут су биле приближних вредности и трендова, те су за анализу узете метрике које су добијене као резултат једног од извршавања. Табела 1 пружа резултате и поређење специфичних аспеката радних оквира, укључујући време генерисања, коришћење ресурса (CPU, RAM, коришћење диска и мреже), величину извештаја, детаљност извештаја, лакоћу интегрисања, лакоћу коришћења и тумачења резултата.

	Maven Surefire	*Allure*
Време генерисања	5s	15s
Величина извештаја	11MB	242MB
Заузеће процесора - системски процеси	81%	89%
Заузеће процесора - кориснички процеси	79%	72%
Потрошња RAM-а	2000MB	3000MB
Network I/O Read (MB/s)	0	19MB/s
Network I/O Write (MB/s)	0	3MB/s
Disk I/O Read (MB/s)	1	0
Disk I/O Write (MB/s)	300MB/s	110MB/s
Кориснички интерфејс	једноставан, функционалан за основне проблеме, без напредних функционалности	атрактиван, модеран, са интерактивним елементима, интерфејс модерних веб апликација
Прилагодљивост	лимитирана прилагодљивост	веома прилагодљив
Лакоћа интеграције	једноставна интеграција као део *Maven build* процеса	захтева инсталацију алата или додатну конфигурацију у зависности од интеграције
Документација	добро документовано	добро документовано

Табела 1. *Резултати метрика и квалитативних особина оба радна оквира за извештавање о тестирању*

5. ЗАКЉУЧАК

Компаративна анализа радних оквира за генерисање извештаја о тестовима *Maven Surefire* и *Allure* истакла је различите предности и мане сваког од алата. *Maven Surefire* је показао брже време генерисања извештаја и мање величине извештаја, што га чини ефикаснијим у смислу брзине и складиштења. Тачније, *Maven Surefire* је генерисао извештаје за 5 секунди са величином од 11 MB, док је *Allure*-у требало 15 секунди и произвео је извештаје од 242 MB. Што се тиче коришћења системских ресурса, *Maven Surefire* је показао нижу употребу RAM-а, од 2000 MB, у поређењу са *Allure*-ових 3000 MB. Међутим, *Allure* је показао веће брзине читања са мреже (19 MB/s) и умерене брзине слања преко мреже (3 MB/s). Поред тога, перформансе брзине писања и читања са диска су варирале, при чему је *Maven Surefire* имао веће брзине писања (300 MB/s), али ниже брзине читања (1 MB/s) од *Allure*-а. Оба оквира су била приближна у коришћењу процесора.

Затим, *Allure* се истакао у дизајну корисничког интерфејса, нудећи визуелно привлачне, интерактивне и прилагодљиве извештаје, који побољшавају могућност корисника да анализира резултате тестова. *Maven Surefire*, иако је био једноставнији, пружио је јасне и детаљне резултате теста. Лакоћа интеграције била је још једна тачка поређења, при чему је *Maven Surefire* био једноставнији за интеграцију у постојеће пројекте засноване на *Maven*-у, док је *Allure* захтевао додатну конфигурацију, али је нудио веће прилагођавање и детаљније карактеристике извештавања. Документација за оба оквира је била добра, иако је *Allure*-ова била опсежнија због ширег скупа функција.

На основу анализе, могу се дати препоруке за коришћење *Maven Surefire* алата када су примарни захтеви брзо генерисање извештаја, ниска потрошња ресурса и једноставна интеграција. *Allure* би било препоручљиво користити у сценаријима где су детаљни и прилагодљиви извештаји од суштинског значаја, а постоји и потреба за интерактивним корисничким интерфејсом како би се олакшала боља анализа резултата теста.

4. ЛИТЕРАТУРА

[1] A. Ramirez, “Maven Surefire Report Plugin – Introduction.” Accessed: Aug. 18, 2024. [Online]. Available: https://maven.apache.org/surefire/maven-surefire-report-plugin/

[2] “Allure Report Docs — Introduction.” Accessed: Aug. 18, 2024. [Online]. Available: https://allurereport.org/docs

[3] “How it works.” Accessed: Aug. 18, 2024. [Online]. Available: https://allurereport.org/docs/how-it-works/

[4] “GitHub Actions documentation,” GitHub Docs. Accessed: Aug. 18, 2024. [Online]. Available: https://docs.github.com/en/actions

[5] “GitHub - catchpoint/workflow-telemetry-action” GitHub. Accessed: Aug. 18, 2024. [Online]. Available: https://github.com/catchpoint/workflow-telemetry-action

Кратка биографија:

Ана Гавриловић рођена је у Шапцу 23. септембра 1999. године. Факултет техничких наука у Новом Саду, студијски програм Рачунарство и аутоматика уписала је 2018. године. Након завршених основних студија, 2022. године, уписала је мастер академске студије из исте области. контакт: ana.gavrilovic247@gmail.com

UDK: 4.41

DOI: https://doi.org/10.24867/30BE46Gavrilovic