

Sumar

1. Crearea unui proiect Maven cu Serenity și JUnit în IntelliJ	2
2. Structura proiectului Maven cu Serenity și JUnit.....	4
3. Alte posibile configurări ale proiectului Maven	4
4. Configurări inițiale și browsere web	5
5. Configurarea webdriver-ului proiecte de testare	6
a. Configurarea webdriver-ului pentru orice proiect de testare	6
b. Configurarea webdriver-ului pentru proiectul de testare.....	6
6. Setarea browser-ului web pentru rularea testelor	7
a. Setarea browser-ului în fișierul de <code>serenity.properties</code>	7
b. Setarea browser-ului în fișierul de <code>pom.xml</code> sau în codul sursă	7
7. Execuția testelor	8
8. Generarea raportului Serenity pentru testele executate.....	9
Varianta 1	9
Varianta 2	10
9. Vizualizarea raportului Serenity	10
10. Data Driven Testing	11

Lista de Figuri

Figure 1. Crearea unui proiect Maven bazat pe Serenity BDD și JUnit	2
Figure 2. Alegerea versiunii de JDK al proiectului Serenity BDD	3
Figure 3. Alegerea versiunii și distribuției JDK care se va descărca	3
Figure 4. Adăugarea tipului de proiect Serenity BDD cu JUnit în lista de tipuri de proiecte Serenity	3
Figure 5. Structura proiectului Maven cu Serenity BDD și JUnit	4
Figure 6. Alegerea browser-ului și a modului de descărcare (manual/automat)	5
Figure 7. Setarea folder-ului pentru webdriver-ele folosite la testare	6
Figure 8. Alegerea browser-ului și setarea folder-ului pentru webdriver în fișierul <code>serenity.properties</code>	7
Figure 9. Setarea browser-ului web Firefox în fișierul <code>pom.xml</code>	7
Figure 10. Crearea unei configurații de rulare a testelor în Serenity BDD	8
Figure 11. Setarea parametrilor pentru comanda <code>verify</code> , cu setarea browser-ului utilizat la execuție	8
Figure 12. Fereastra Maven Projects și comenzile Maven	9
Figure 13. Fereastra Maven Projects și opțiunea Execute Maven Goal - generarea raportului Serenity	9
Figure 14. Fereastra de comenzi Maven pentru generarea raportului de testare	10
Figure 15. Fereastra Terminal cu execuția comenzii de generare a raportului de testare	10
Figure 16. Vizualizarea raportului Serenity	10
Figure 17. Crearea fișierului cu date de test	11
Figure 18. Clasă de test parametrizată, folosind un fișier <code>.csv</code>	11
Figure 19. Raportul Serenity pentru testele din fișierul <code>.csv</code>	12

1. Crearea unui proiect Maven cu Serenity și JUnit în IntelliJ

1. în meniul **File** ---> **New** ---> **Project**;
2. se selectează din tipul de proiect bazat pe Maven ---> **Maven Archetype** (vezi Figure 1);
3. se indică numele proiectului, e.g., AutoDemo;
4. se precizează folderul în care se salvează proiectul;
5. se alege o versiune pentru JDK; dacă s-a lucrat anterior cu *Java 23* se alege versiunea 1.8 sau 14. Dacă aceasta nu este instalată deja, atunci
 - se alege opțiunea *Download JDK* (vezi Figure 2) și apoi
 - se selectează versiunea, e.g., **1.8**. și distribuția, e.g., **Amazon Corretto 1.8** (vezi Figure 3); se păstrează folderul implicit pentru descărcarea noii versiuni de JDK.
6. se indică sursa de la care se va aduce șablonul de proiect Maven, alegând *catalogul*, i.e., **Maven Central**.
7. se alege tipul de proiect din lista *Archetypes* tipul de proiect este SerenityBDD cu JUnit, identificat prin
 - framework: **net.serenity-bdd**
 - tip de proiect: **serenity-junit-archetype**
 - dacă tipul de proiect dorit nu există, atunci el poate fi adăugat folosind opțiunea **Add...** și completând informațiile (vezi Figure 4)
 - *GroupId*: **net.serenity-bdd**
 - *ArtifactId*: **serenity-junit-archetype**
 - *Version*: **2.3.2**
 - apoi **Add**;
 - Lista de proiecte Maven care folosesc Serenity poate fi accesată la acest [link](#), iar lista de versiuni disponibile pentru un proiect SerenityBDD + JUnit poate fi accesată la acest [link](#).

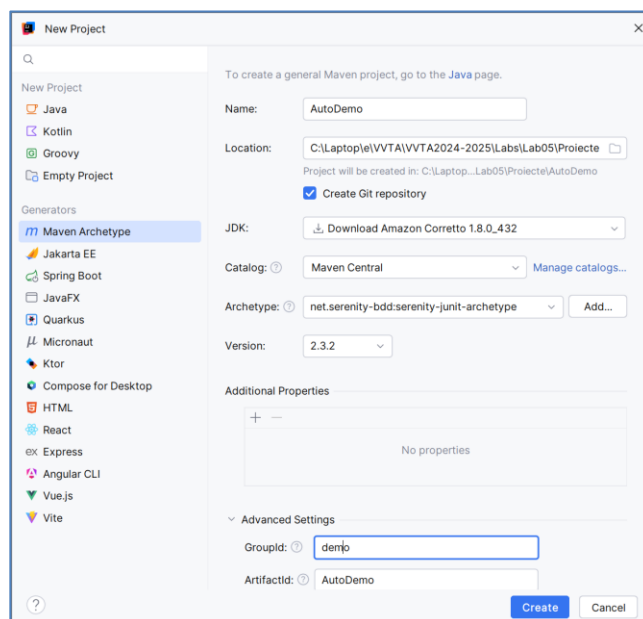


Figure 1. Crearea unui proiect Maven bazat pe Serenity BDD și JUnit

8. se completează numele pachetului root, i.e., **demo**,
9. se completează numele proiectului, i.e., **AutoDemo**,

10. apoi **Create** pentru a crea proiectul SerenityBDD cu JUnit.

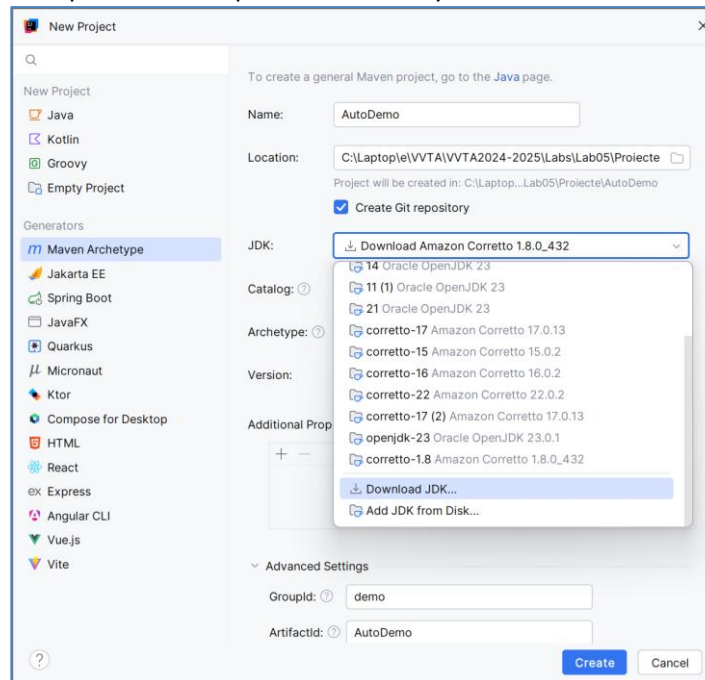


Figure 2. Alegerea versiunii de JDK al proiectului Serenity BDD

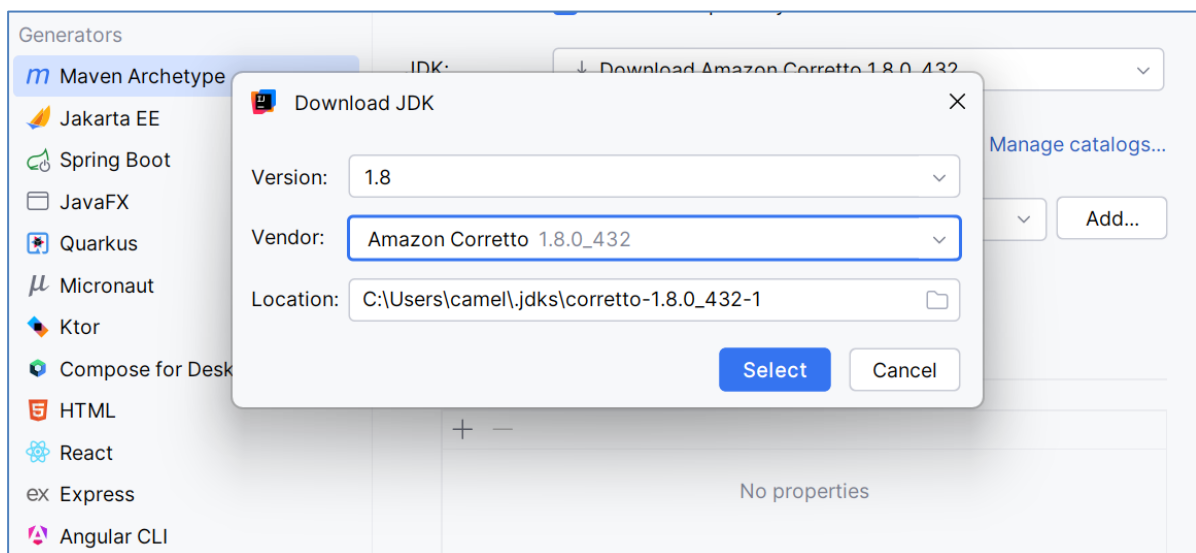


Figure 3. Alegerea versiunii si distribuției JDK care se va descărca

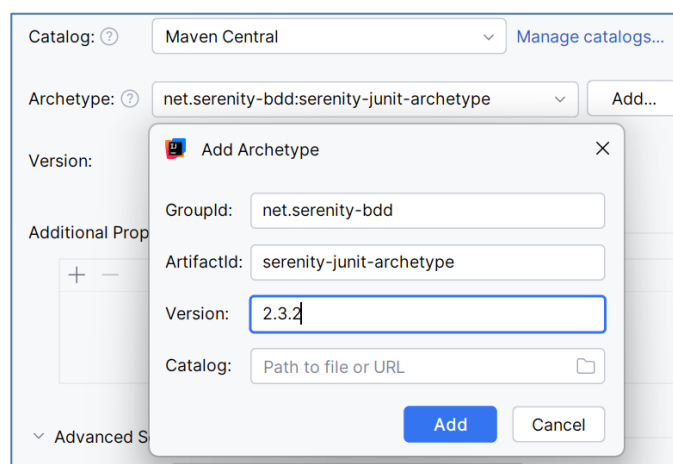


Figure 4. Adăugarea tipului de proiect Serenity BDD cu JUnit în lista de tipuri de proiecte Serenity

2. Structura proiectului Maven cu Serenity și JUnit

1. după creare, proiectul are o structură similară cu cea a unui proiect Maven care conține **doar** pachetul **test/java** (vezi Figure 5);
2. particularitățile proiectului Maven Serenity cu JUnit sunt reprezentate de:
 - pachetul **main/java** nu este generat deoarece aplicația care urmează să fie testată este accesată prin intermediul unei adrese web, așadar nu este necesar să adăugăm cod sursă într-un asemenea pachet;
 - pachetul **test/java** conține câteva subpachete predefinite, i.e., **features/search**, **pages**, **steps**, cât și clase generate care implementează șablonul pentru testarea aplicațiilor web **Page Object Model**;

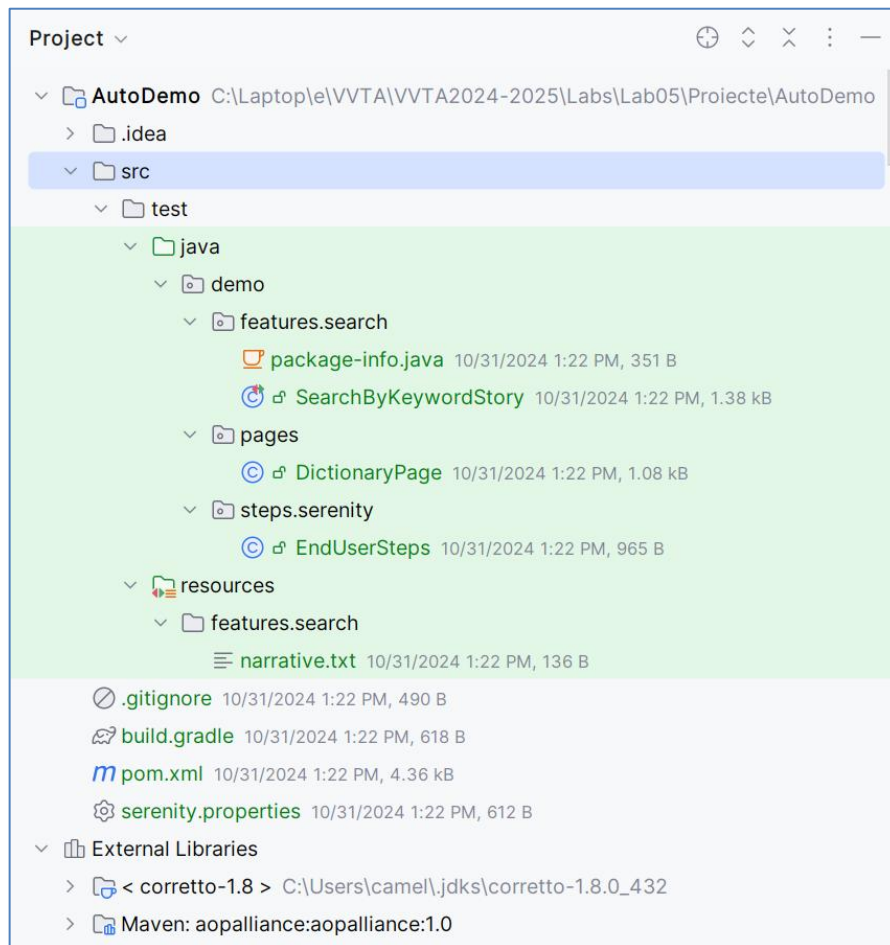


Figure 5. Structura proiectului Maven cu Serenity BDD și JUnit

3. Alte posibile configurări ale proiectului Maven

La crearea proiectului Maven, în funcție de versiunea Java instalată, cât și de alte configurări și setări existente, mai pot fi necesare și următoarele configurări în fișierul **pom.xml**:

1. este posibil ca la descărcarea resurselor asociate proiectului Serenity BDD să fie necesară modificarea protocolului utilizat (din HTTP în HTTPS) în elementul care descrie adresa de acces la repository-ul Maven Central folosit, e.g.,
`<url>https://jcenter.bintray.com</url>`
2. anumite versiuni de Java SE nu conțin implicit JAXB APIs; acestea pot fi incluse manual în fișierul **pom.xml** ca dependențe, în secțiunea `<dependencies>...`
`</dependencies>`

```

<dependency>
  <groupId>javax.xml.bind</groupId>
  <artifactId>jaxb-api</artifactId>
  <version>2.3.0</version>
</dependency>

<dependency>
  <groupId>com.sun.xml.bind</groupId>
  <artifactId>jaxb-impl</artifactId>
  <version>2.3.0</version>
</dependency>

<dependency>
  <groupId>javax.activation</groupId>
  <artifactId>activation</artifactId>
  <version>1.1.1</version>
</dependency>

<dependency>
  <groupId>com.sun.xml.bind</groupId>
  <artifactId>jaxb-core</artifactId>
  <version>2.3.0</version>
</dependency>

```

Așadar, aceste configurări sunt necesare doar dacă la rularea testelor incluse implicit în proiect apar erori. Altfel, aceste modificări nu sunt obligatorii în fișierul *pom.xml*.

4. Configurări inițiale și browsere web

1. La crearea proiectului Maven cu Serenity și JUnit, browser-ul web Chrome este considerat browser-ul implicit pentru rularea testelor, iar fișierul webdriver corespunzător este descărcat automat.
2. În acest tutorial se vor detalia pașii pentru configurarea manuală a webdriver-ului, i.e., alegerea unui browser particular, a unei versiuni a acestuia și a folder-ului în care se salvează webdriver-ul corespunzător.
3. Pentru alegerea browser-ului și configurarea manuală a webdriver-ului, în fișierul **serenity.properties** se setează opțiunile (vezi Figure 6):
 - `webdriver.driver = firefox` sau `chrome` sau alt browser
 - `webdriver.autodownload = false`, inițial valoarea este `true`
 - `headless.mode = false`, inițial valoarea este `true`, indicând modul de execuție, cu sau fără vizualizarea browser-ului.

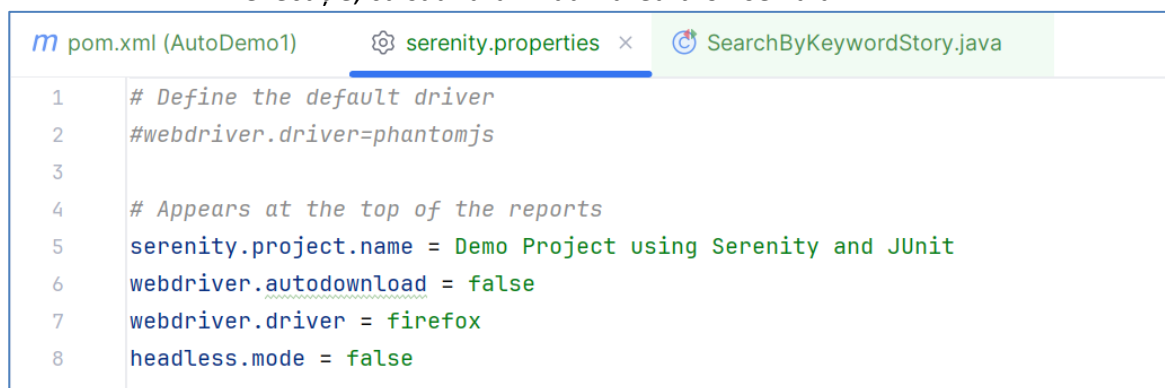


Figure 6. Alegerea browser-ului și a modului de descărcare (manual/automat)

4. driverele pentru browser-ele web folosite la testare se pot descărca manual de la adresele:
 - **Firefox:**
 - adresa web: <https://github.com/mozilla/geckodriver/releases>;

- indiferent de versiunea browser-ului Firefox instalată, se folosește un singur webdriver, care depinde de sistemul de operare, e.g., **geckodriver-v0.35.0-win64.zip**.

• **Chrome:**

- adresa web: <https://googlechromelabs.github.io/chrome-for-testing/>;
- webdriver-ul depinde de versiunea browser-ului și de sistemul de operare, e.g., <https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/win32/chromedriver-win32.zip>.

chromedriver	win32	https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.91/win32/chromedriver-win32.zip	200
--------------	-------	---	-----

- se recomandă verificarea versiunii browser-ului Chrome și descărcarea variantei *Stable* a webdriver-ului.

5. Configurarea webdriver-ului proiecte de testare

a. Configurarea webdriver-ului pentru orice proiect de testare

1. Webdriver-ele se descarcă, se dezarchivează, iar fișierele .exe se salvează într-un folder, e.g., **c:\drivers**, de unde pot fi folosite ulterior de orice proiect de testare.
2. în variabila de mediu **Path** se adaugă calea către folder-ul care conține webdriver-ele pentru browser-ele web, i.e., **c:\drivers** (vezi Figure 7), **apoi se restartează sistemul de operare**.

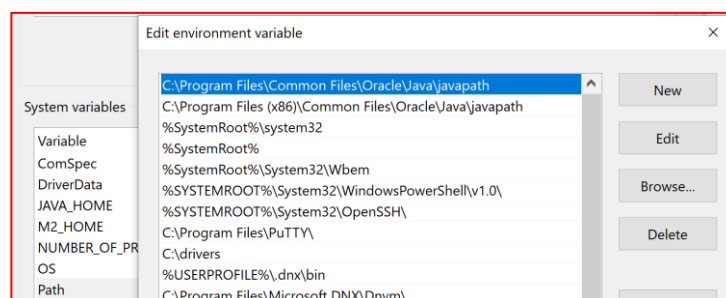


Figure 7. Setarea folder-ului pentru webdriver-ele folosite la testare

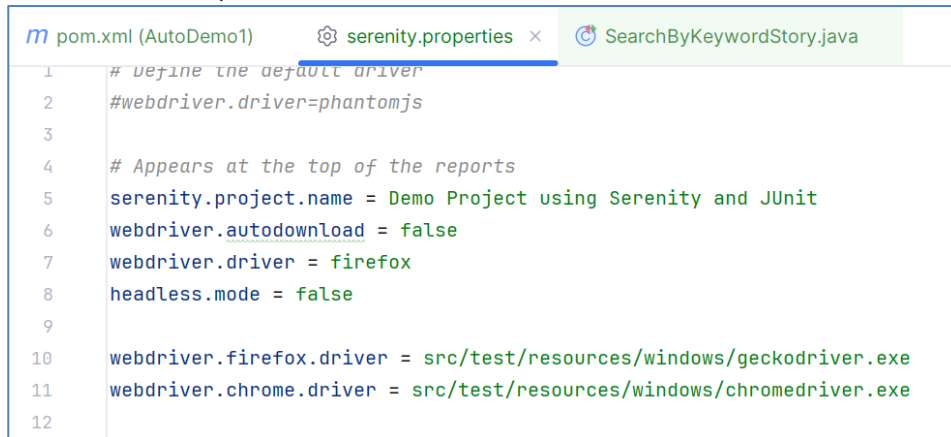
b. Configurarea webdriver-ului pentru proiectul de testare

1. Webdriver-ele se descarcă, se dezarchivează, iar fișierele .exe se salvează într-un folder în cadrul proiectului de testare, e.g., **src/test/resources**, în care se vor crea, după caz, subfolder-e pentru sistemul de operare și/sau browser-ul folosit (vezi Figure 8).
2. În fișierul `serenity.properties` se indică în proprietățile driver corespunzătoare browser-elor fișierul ce reprezintă webdriver-ul. De exemplu:
 - `webdriver.firefox.driver = src/test/resources/windows/geckodriver.exe`
 - `webdriver.chrome.driver = src/test/resources/windows/chromedriver.exe`

6. Setarea browser-ului web pentru rularea testelor

a. Setarea browser-ului în fișierul de `serenity.properties`

1. Pentru alegerea browser-ului, în fișierul `serenity.properties` se setează opțiunile (vezi Figure 8):
 - `webdriver.driver = firefox` sau `chrome` sau alt browser
 - `headless.mode = false`, inițial valoarea este `true`, indicând modul de execuție, cu sau fără vizualizarea browser-ului.



```

1 # Define the default driver
2 #webdriver.driver=phantomjs
3
4 # Appears at the top of the reports
5 serenity.project.name = Demo Project using Serenity and JUnit
6 webdriver.autodownload = false
7 webdriver.driver = firefox
8 headless.mode = false
9
10 webdriver.firefox.driver = src/test/resources/windows/geckodriver.exe
11 webdriver.chrome.driver = src/test/resources/windows/chromedriver.exe
12

```

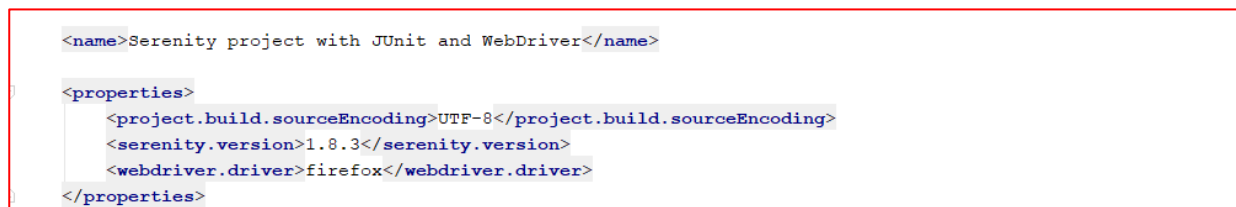
Figure 8. Alegerea browser-ului și setarea folder-ului pentru webdriver în fișierul `serenity.properties`

b. Setarea browser-ului în fișierul de `pom.xml` sau în codul sursă

În funcție de versiunea de proiect Serenity BDD, e.g., 1.8.4, 2.0.81 sau 2.3.2, indicarea browser-ului folosit pentru rularea testelor se poate face diferit.

Pentru versiunea 1.8.3/1.8.4:

1. fișierul `pom.xml` generat conține specificarea referitoare la browser-ul folosit la rularea testelor, i.e., implicit **firefox** (vezi Figure 9);
2. pentru schimbarea browser-ului se modifică în fișierul `pom.xml` tipul driverului, i.e., **chrome** sau **firefox**;



```

<name>Serenity project with JUnit and WebDriver</name>

<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <serenity.version>1.8.3</serenity.version>
  <webdriver.driver>firefox</webdriver.driver>
</properties>

```

Figure 9. Setarea browser-ului web Firefox în fișierul `pom.xml`

Pentru versiunea 2.0.81 sau 2.3.2:

Varianta 1: folosind adnotarea `@Managed`

1. în clasa cu teste, la adnotarea `@Managed` se adăuga numele browser-ului utilizat la rulare; astfel:
 - rulare cu browser-ul Chrome:


```
@Managed(uniqueSession = true, driver="chrome")
public WebDriver webdriver;
```
 - rulare cu browser-ul Firefox:


```
@Managed(uniqueSession = true, driver="firefox")
public WebDriver webdriver;
```
 - rulare cu browser-ul implicit (Firefox):


```
@Managed(uniqueSession = true)
public WebDriver webdriver;
```

Varianta 2: folosind o configurație de rulare

2. în fereastra de comenzi Maven, pentru comanda **verify** se creează o configurație de rulare: click dreapta pe **verify** ---> **Create [configuration]** (vezi Figure 10);
3. pentru comanda **verify** se utilizează ca parametri **context** și **webdriver.driver** (vezi Figure 11):

```
verify -Dcontext=chrome -Dwebdriver.driver=chrome -f pom.xml
```

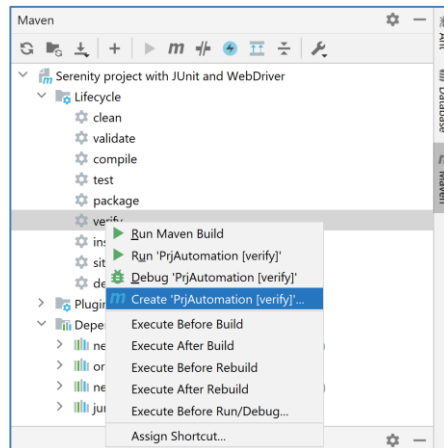


Figure 10. Crearea unei configurații de rulare a testelor în Serenity BDD

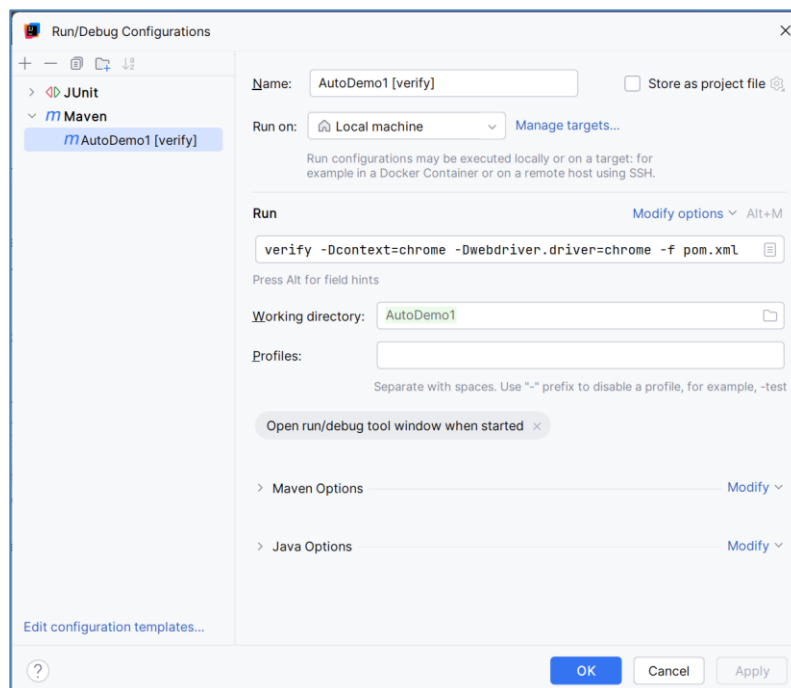


Figure 11. Setarea parametrilor pentru comanda **verify**, cu setarea browser-ului utilizat la execuție

7. Execuția testelor

1. click dreapta pe un test sau o clasă de test în **Project Explorer**, e.g., **SearchByKeywordStory**;
2. se selectează **Run SearchByKeywordStory**;

8. Generarea raportului *Serenity* pentru testele executate

Varianta 1

1. execuția testelor nu implică și generarea raportului de testare;
2. din meniul **View** ---> **Tool Windows** ---> **Maven Projects** se deschide fereastra proiectelor gestionate cu Maven (vezi Figure 12);
3. pentru execuția tuturor testelor sau execuția individuală a unui test sau clase de teste (vezi Figure 11, Figure 12 sau Secțiunea Execuția testelor) se folosește comanda **verify**;
4. din meniul ferestrei proiectelor gestionate cu Maven se alege opțiunea **Execute Maven Goal** (vezi Figure 13);
5. în fereastra de comenzi Maven se completează comanda `mvn serenity:aggregate` , apoi **<enter>** (vezi Figure 14);
6. raportul generat va fi salvat în folderul proiectului în `\target\site\serenity`;

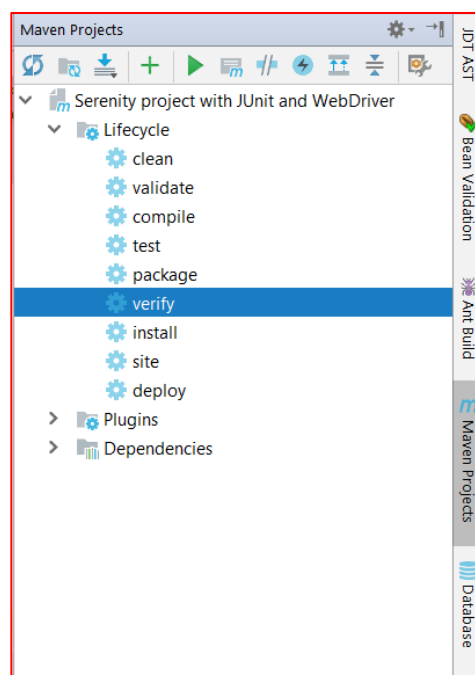


Figure 12. Fereastra Maven Projects și comenzile Maven

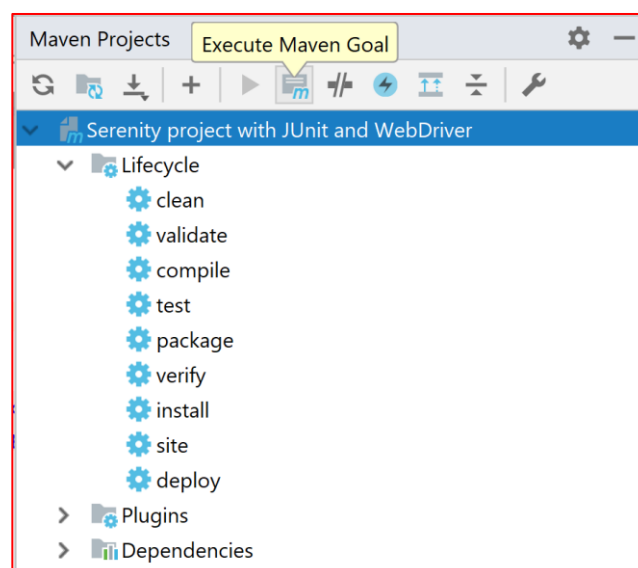


Figure 13. Fereastra Maven Projects și opțiunea Execute Maven Goal - generarea raportului Serenity

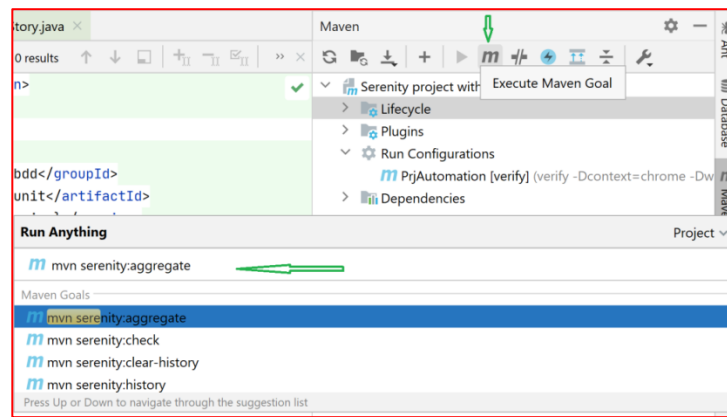


Figure 14. Fereastra de comenzi Maven pentru generarea raportului de testare

Varianta 2

1. **presupune ca Maven sa fie instalat local** și folderul în care Maven este instalat să fie inclus în variabila de mediu **Path**;
2. din meniul **View ---> Tool Windows ---> Terminal** se deschide fereastra care permite execuția comenzilor din linia de comandă;
3. în această fereastră se execută comanda Maven:
`mvn serenity: aggregate <enter>` (vezi Figure 15);
4. raportul generat va fi salvat în folderul proiectului în `\target\site\serenity`;

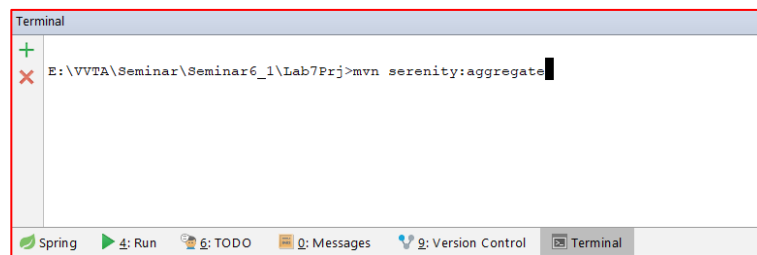


Figure 15. Fereastra Terminal cu execuția comenzii de generare a raportului de testare

9. Vizualizarea raportului Serenity

- din folderul `\target\site\serenity` se încarcă într-un browser web fișierul `index.html` (vezi Figure 16).

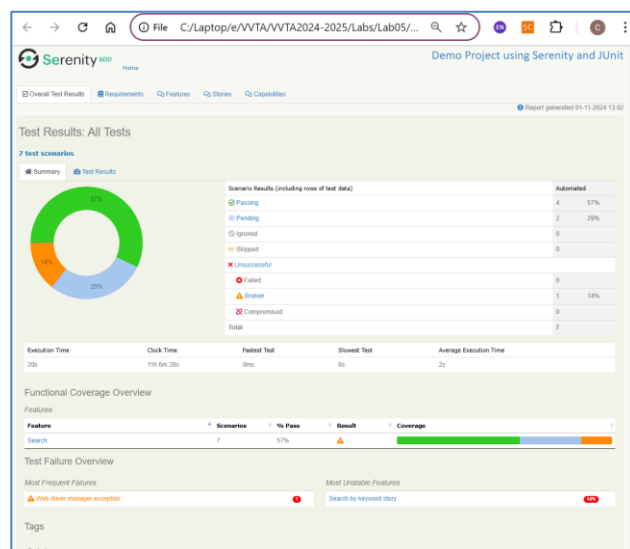
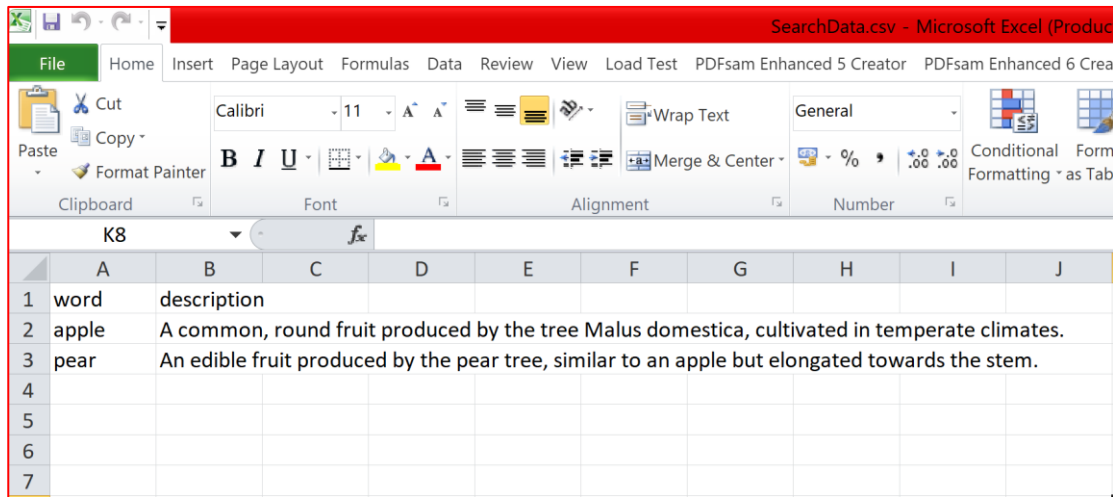


Figure 16. Vizualizarea raportului Serenity

10.Data Driven Testing

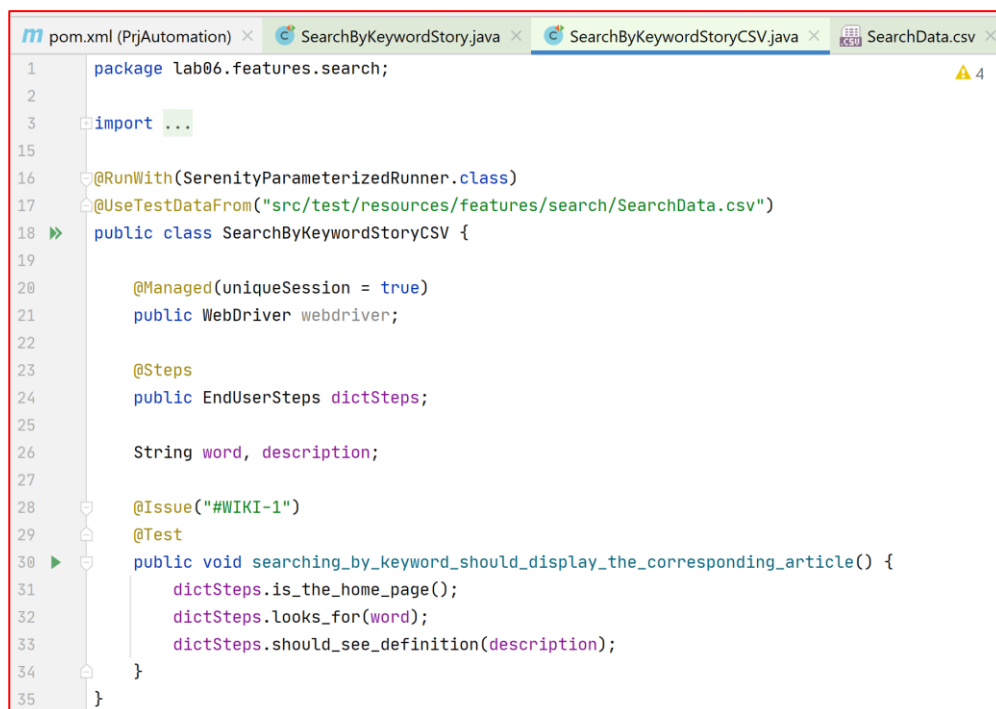
1. se creează fișierul **SearchData.csv** cu date de test (vezi Figure 17);
2. prima linie din fișier indică structura tabelului cu date de intrare;
3. următoarele linii conțin date de intrare pentru cazuri de testare individuale;



	A	B	C	D	E	F	G	H	I	J
1	word	description								
2	apple	A common, round fruit produced by the tree Malus domestica, cultivated in temperate climates.								
3	pear	An edible fruit produced by the pear tree, similar to an apple but elongated towards the stem.								
4										
5										
6										
7										

Figure 17. Crearea fișierului cu date de test

4. fișierul **SearchData.csv** se salvează în folderul **src/test/resources**;
5. clasa de test va fi executată folosind un *runner parametrizat*;
6. clasa de test reprezintă un șablon de test care va fi folosit pentru execuția fiecărui test pentru care se vor prelua datele de intrare din fiecare linie a fișierului .csv, fișierul fiind dat ca parametru (vezi Figure 18);
7. rularea testelor se realizează ca în Secțiunea 7;
8. generarea raportului Serenity se realizează ca în Secțiunea 8;
9. vizualizarea raportului Serenity indică execuția aceluiași test cu date de intrare diferite, preluate din fișierul .csv, dat ca parametru (vezi Figure 19).



```

1 package lab06.features.search;
2
3 import ...
4
5
6
7
8
9
10
11
12
13
14
15
16 @RunWith(SerenityParameterizedRunner.class)
17 @UseTestDataFrom("src/test/resources/features/search/SearchData.csv")
18 public class SearchByKeywordStoryCSV {
19
20     @Managed(uniqueSession = true)
21     public WebDriver webdriver;
22
23     @Steps
24     public EndUserSteps dictSteps;
25
26     String word, description;
27
28     @Issue("#WIKI-1")
29     @Test
30     public void searching_by_keyword_should_display_the_corresponding_article() {
31         dictSteps.is_the_home_page();
32         dictSteps.looks_for(word);
33         dictSteps.should_see_definition(description);
34     }
35 }

```

Figure 18. Clasă de test parametrizată, folosind un fișier .csv

Examples:

#	Word	Description
1	apple	A common, round fruit produced by the tree <i>Malus domestica</i> , cultivated in temperate climates.
2	pear	An edible fruit produced by the pear tree, similar to an apple but elongated towards the stem.



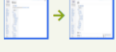

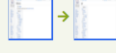



Steps	Screenshots	Outcome	⌚
<div>+</div> Searching by keyword should display the corresponding article		SUCCESS	22.25s
<div>✓</div> Is the home page		SUCCESS	8.76s
<div>+</div> Looks for: apple		SUCCESS	11.2s
<div>✓</div> Should see definition: A common, round fruit produced by the tree <i>Malus domestica</i> , cultivated in temperate climates.		SUCCESS	2.3s
<div>+</div> Searching by keyword should display the corresponding article		SUCCESS	18.22s
<div>✓</div> Is the home page		SUCCESS	14.2s
<div>+</div> Looks for: pear		SUCCESS	3.59s
<div>✓</div> Should see definition: An edible fruit produced by the pear tree, similar to an apple but elongated towards the stem.		SUCCESS	0.44s
		SUCCESS	40.48s

Figure 19. Raportul Serenity pentru testele din fișierul .csv