

SELENIUM

Автоматизированное тестирование Web сайтов

Практические примеры 2024

Содержание

Введение	2
Установка и запуск Selenium Grid	4
Создание проекта Selenium (JavaScript) с простым автотестом	6
Создание проекта Selenium (Java) с простым автотестом	8
Практика применения JUnit	12
Запуск автотестов с помощью Maven	16
Практика применения TestNG	18
Создание проекта Selenium (Python) с простым автотестом	21
Практика применения Unittest	26
Практика применения PyTest	
Фреймворк WebdriverIO	
Фреймворк Cucumber	
Фреймворк Selenide	
Фреймворк Robot	
Фреймворк Codeception	
Отчет Allure Report	
Отчет Report Portal	
Jenkins	
Docker	
Нагрузочное тестирование с помощью Jmeter	
Тестирование API с помощью Bruno	

Введение

В данном документе описана практика разработки автоматизированных тестов на основе технологии Selenium с использованием специальных фреймворков таких как: Cucumber, Selenide, Robot, WebDriverIO, Codeception.

Для создания автотестов будут применяться технологии модульного тестирования JUnit, TestNG, Unittest, PyTest, PHPUnit.

В демонстрационных примерах отражена полноценная разработка автотестов через паттерны PageObject, StepsObject.

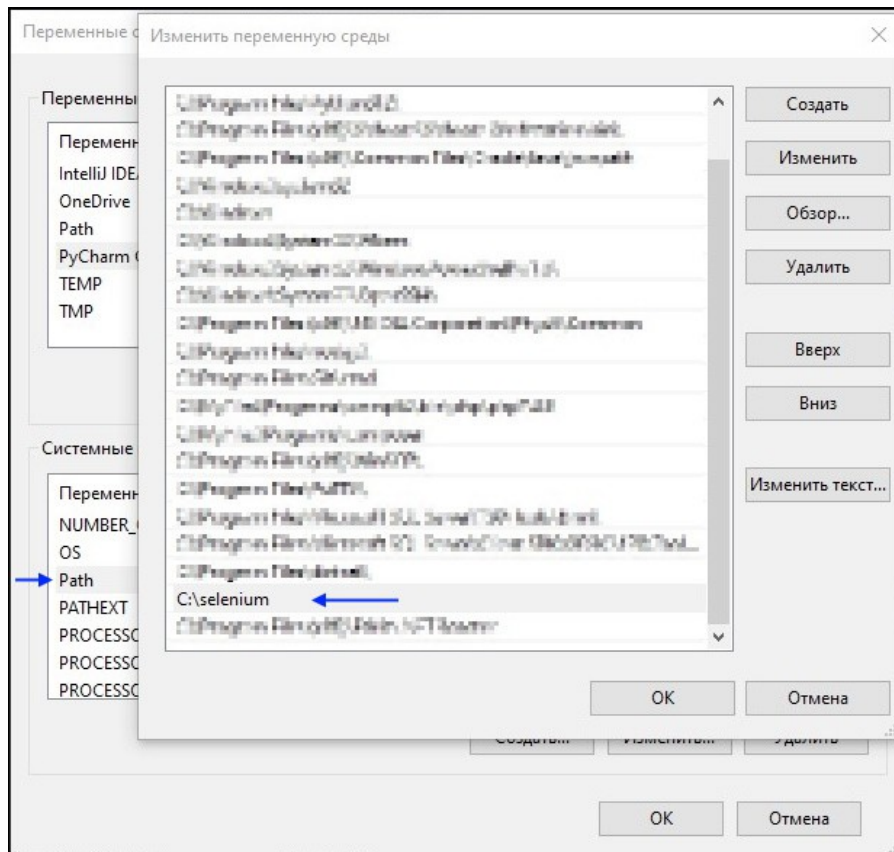
В дополнение практических занятий будет рассмотрено нагрузочное тестирование с помощью JMeter.

Полезные ссылки

- Официальный сайт Selenium:
<https://www.selenium.dev/>
- Документация Selenium
<https://www.selenium.dev/documentation/overview/>
- Быстрый старт Selenium
https://www.selenium.dev/documentation/grid/getting_started/
- Скачать Selenium Server (Grid):
<https://www.selenium.dev/downloads/>
- Официальная страница Chrome драйвер
<https://googlechromelabs.github.io/chrome-for-testing/#stable>
- Официальная документация The Selenium Browser Automation Project
<https://www.selenium.dev/documentation/>
- Официальная страница Visual Studio Code
<https://code.visualstudio.com/>
- Официальная страница NodeJS
<https://nodejs.org/>
- Официальная страница IntelliJ IDEA Community Edition
<https://www.jetbrains.com/idea/download/other.html>
- Официальная страница Maven
<https://mvnrepository.com/>
- Официальная страница junit5
<https://junit.org/junit5/>
- Официальная страница TestNG
<https://testng.org/>
- Официальная страница Java SE Development Kit 11
<https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html>
- Официальная страница PyCharm Community Edition
<https://www.jetbrains.com/pycharm/download/other.html>
- Официальная страница Python
<https://www.python.org/>

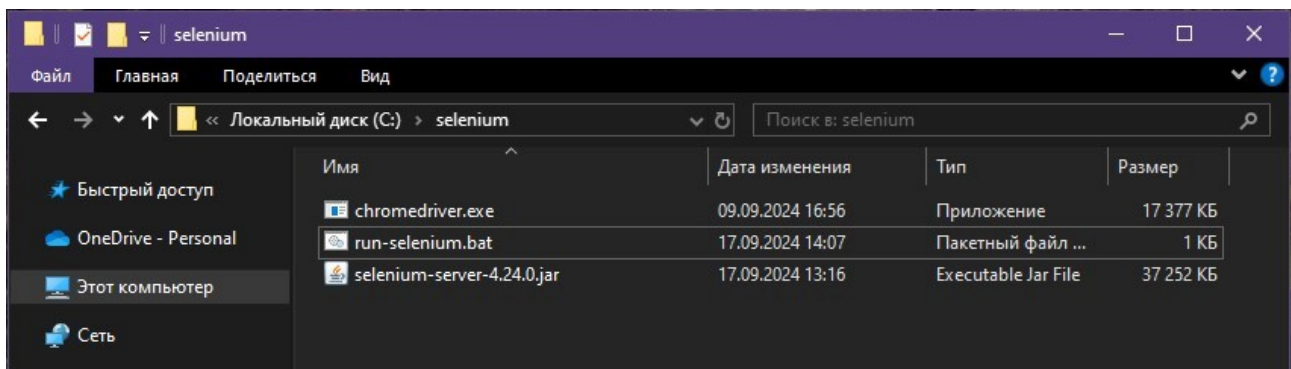
Установка и запуск Selenium Grid

1. Скачать и установить [Java SE Development Kit 11](#)
2. Скачать и установить браузер [Chrome](#)
3. Скачать драйвер [ChromeDriver](#)
4. Скачать jar файл [Selenium Server](#) или с [github](#)
5. Создать папку C:\selenium\ в которую скопировать файлы:
selenium-server-4.24.0.jar, chromedriver.exe
6. В переменной **Path** прописать путь к папке C:\selenium
(Панель управления > Система > Дополнительные параметры системы > Переменные среды)



7. В папке C:\selenium\ создать файл run-selenium.bat в котором написать:

```
cd C:\selenium
java -jar selenium-server-4.24.0.jar standalone
```



8. Запустить файл run-selenium.bat

```
C:\Windows\System32\cmd.exe - run-selenium.bat
Microsoft Windows [Version 10.0.19045.4894]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\selenium>run-selenium.bat

C:\selenium>cd C:\selenium

C:\selenium>java -jar selenium-server-4.24.0.jar standalone
14:08:20.648 INFO [LoggingOptions.configureLogEncoding] - Using the system default encoding
14:08:20.656 INFO [OpenTelemetryTracer.createTracer] - Using OpenTelemetry for tracing
14:08:22.215 INFO [NodeOptions.getSessionFactories] - Detected 4 available processors
14:08:22.216 INFO [NodeOptions.discoverDrivers] - Looking for existing drivers on the PATH.
14:08:22.217 INFO [NodeOptions.discoverDrivers] - Add '--selenium-manager true' to the startup command to setup drivers automatically.
14:08:28.219 WARN [SeleniumManager.lambda$runCommand$1] - Exception managing chrome: Unable to discover proper chromedriver version in offline mode
14:08:28.490 WARN [SeleniumManager.lambda$runCommand$1] - Unable to discover proper msedgedriver version in offline mode
14:08:28.755 WARN [SeleniumManager.lambda$runCommand$1] - Unable to discover proper geckodriver version in offline mode
14:08:29.080 WARN [SeleniumManager.lambda$runCommand$1] - Unable to discover proper IEDriverServer version in offline mode
14:08:29.096 INFO [NodeOptions.report] - Adding Edge for {"browserName": "MicrosoftEdge", "platformName": "Windows 10"} 4 times
14:08:29.097 INFO [NodeOptions.report] - Adding Chrome for {"browserName": "chrome", "platformName": "Windows 10"} 4 times
14:08:29.098 INFO [NodeOptions.report] - Adding Internet Explorer for {"browserName": "internet explorer", "platformName": "Windows 10"} 1 times
14:08:29.099 INFO [NodeOptions.report] - Adding Firefox for {"browserName": "firefox", "platformName": "Windows 10"} 4 times
14:08:29.134 INFO [Node.<init>] - Binding additional locator mechanisms: relative
14:08:29.152 INFO [GridModel.setAvailability] - Switching Node 4eb0dc24-9f81-4a19-8b01-a164bf8911aa (uri: http://192.168.132.1:4444) from DOWN to UP
14:08:29.153 INFO [LocalDistributor.add] - Added node 4eb0dc24-9f81-4a19-8b01-a164bf8911aa at http://192.168.132.1:4444. Health check every 120s
14:08:29.614 INFO [Standalone.execute] - Started Selenium Standalone 4.24.0 (revision 748ffc9bc3): http://192.168.132.1:4444
```

9. Чтобы остановить Selenium нажмите в консоли Ctrl+C и затем Y

Создание проекта Selenium (JavaScript) с простым автотестом

1. Скачать и установить [Visual Studio Code](#)

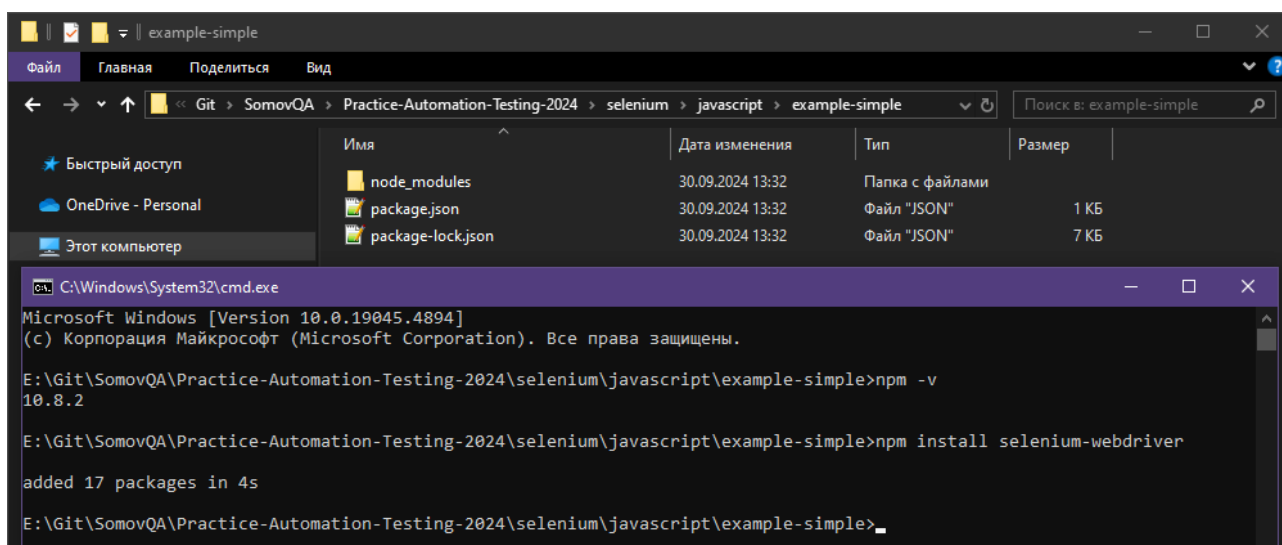
2. Скачать и установить [NodeJS](#)

3. Открыть консоль и проверить работу NodeJS с помощью команды:

```
npm -v
```

4. Создать папку проекта example-simple и в корне этой папки выполнить в консоли команду:

```
npm install selenium-webdriver
```



5. Открыть папку в редакторе Visual Studio Code

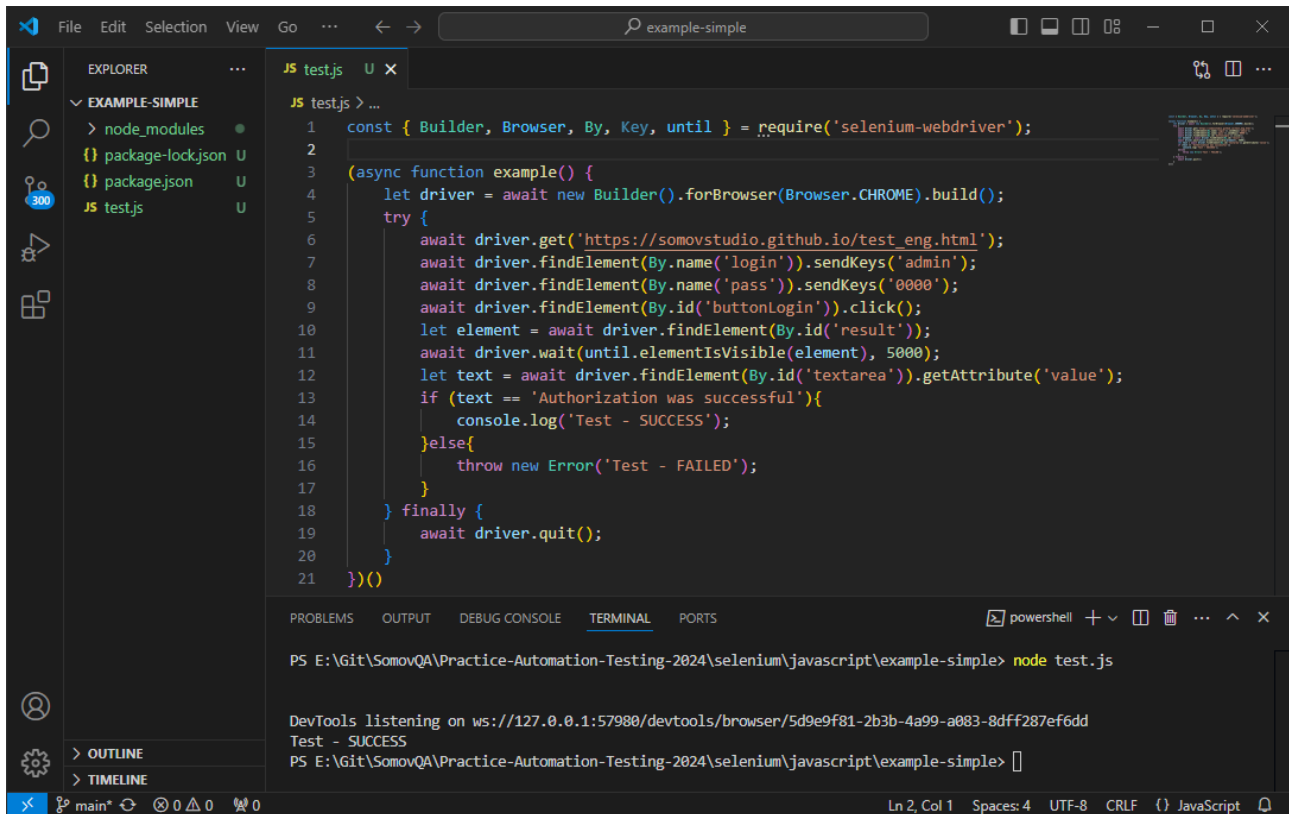
6. Создать файл test.js и описать автотест следующим образом

```
(async function example() {  
  let driver = await new Builder().forBrowser(Browser.CHROME).build();  
  try {  
    await driver.get('https://somovstudio.github.io/test_eng.html');  
    await driver.findElement(By.name('login')).sendKeys('admin');  
    await driver.findElement(By.name('pass')).sendKeys('0000');  
    await driver.findElement(By.id('buttonLogin')).click();  
    let element = await driver.findElement(By.id('result'));  
    await driver.wait(until.elementIsVisible(element), 5000);  
    let text = await driver.findElement(By.id('textarea')).getAttribute('value');  
    if (text == 'Authorization was successful'){  
      console.log('Test - SUCCESS');  
    }else{  
      throw new Error('Test - FAILED');  
    }  
  } finally {  
    await driver.quit();  
  }  
})()
```

7. Запустить автотест командой

```
node test.js
```

8. Результат автотеста в консоли

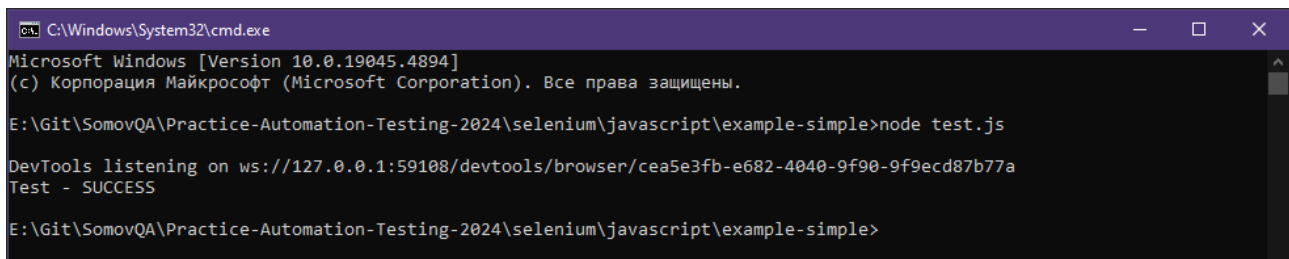


The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane shows a project named 'EXAMPLE-SIMPLE' with files 'package-lock.json', 'package.json', and 'test.js'. The main editor displays the content of 'test.js', which is a JavaScript file using Selenium WebDriver to automate a login process. The code includes imports for Selenium WebDriver, a function 'example' that performs the login steps, and a test runner function 'test' that calls 'example' and logs the result. The terminal at the bottom shows the command 'node test.js' being executed, followed by the output 'Test - SUCCESS'.

```
1 const { Builder, Browser, By, Key, until } = require('selenium-webdriver');
2
3 (async function example() {
4   let driver = await new Builder().forBrowser(Browser.CHROME).build();
5   try {
6     await driver.get('https://somovstudio.github.io/test_eng.html');
7     await driver.findElement(By.name('login')).sendKeys('admin');
8     await driver.findElement(By.name('pass')).sendKeys('0000');
9     await driver.findElement(By.id('buttonLogin')).click();
10    let element = await driver.findElement(By.id('result'));
11    await driver.wait(until.elementIsVisible(element), 5000);
12    let text = await driver.findElement(By.id('textarea')).getAttribute('value');
13    if (text == 'Authorization was successful'){
14      console.log('Test - SUCCESS');
15    }else{
16      throw new Error('Test - FAILED');
17    }
18  } finally {
19    await driver.quit();
20  }
21 })()
```

```
PS E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\javascript\example-simple> node test.js

DevTools listening on ws://127.0.0.1:57980/devtools/browser/5d9e9f81-2b3b-4a99-a083-8dff287ef6dd
Test - SUCCESS
PS E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\javascript\example-simple>
```



The screenshot shows a Windows Command Prompt window. It displays the command 'node test.js' being executed in the directory 'E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\javascript\example-simple'. The output shows 'DevTools listening on ws://127.0.0.1:59108/devtools/browser/cea5e3fb-e682-4040-9f90-9f9ecd87b77a' followed by 'Test - SUCCESS'.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.4894]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\javascript\example-simple>node test.js

DevTools listening on ws://127.0.0.1:59108/devtools/browser/cea5e3fb-e682-4040-9f90-9f9ecd87b77a
Test - SUCCESS

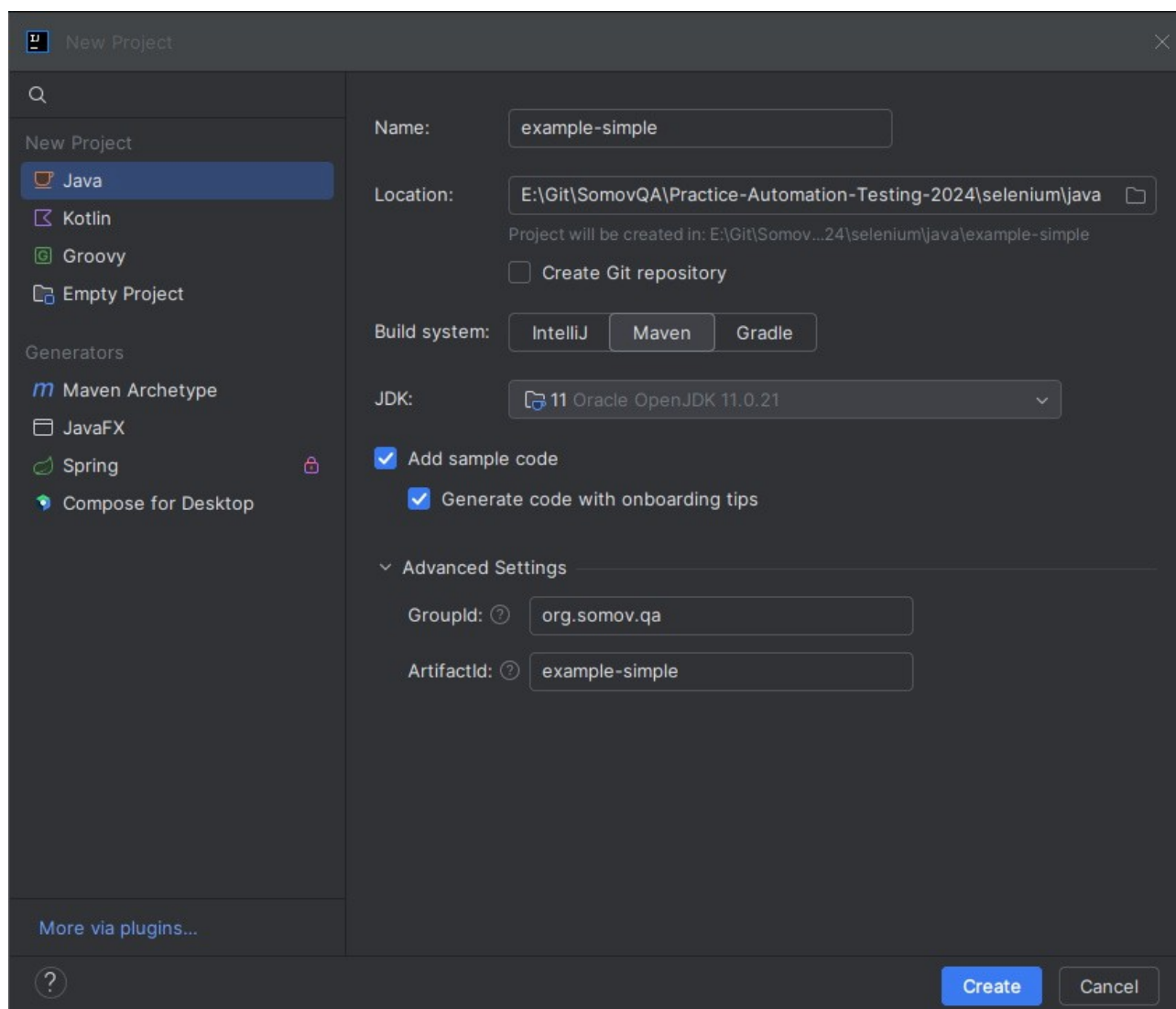
E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\javascript\example-simple>
```

Полезные ссылки:

- Официальная документация The Selenium Browser Automation Project
<https://www.selenium.dev/documentation/>
- Документация API Docs
<https://www.selenium.dev/selenium/docs/api/javascript/index.html>
- Официальная страница Visual Studio Code
<https://code.visualstudio.com/>
- Официальная страница NodeJS
<https://nodejs.org/>

Создание проекта Selenium (Java) с простым автотестом

1. Скачать и установить [IntelliJ IDEA Community Edition](#)
2. Скачать и установить [Java SE Development Kit 11](#)
3. В редакторе IntelliJ IDEA создаем новый проект example-simple выбрав Maven и SDK: Oracle OpenJDK 11



4. Подключить библиотеку Selenium к проекту в файле pom.xml
ссылка: <https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>

```
<dependencies>
  <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.25.0</version>
  </dependency>
</dependencies>
```


5. Выполнить загрузку библиотеки в файле pom.xml

```
17 <dependencies>
18 <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium
19 <dependency>
20 <groupId>org.seleniumhq.selenium</groupId>
21 <artifactId>selenium-java</artifactId>
22 <version>4.24.0</version>
23 </dependency>
24 </dependencies>
25
```

Load Maven Changes Ctrl+Shift+O
Maven project structure has been changed. Load changes into IntelliJ IDEA to make it work correctly.

```
m pom.xml (example-simple) x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5   <modelVersion>4.0.0</modelVersion>
6
7   <groupId>org.somov.qa</groupId>
8   <artifactId>example-simple</artifactId>
9   <version>1.0-SNAPSHOT</version>
10
11   <properties>
12     <maven.compiler.source>11</maven.compiler.source>
13     <maven.compiler.target>11</maven.compiler.target>
14     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15   </properties>
16
17   <dependencies>
18     <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
19     <dependency>
20       <groupId>org.seleniumhq.selenium</groupId>
21       <artifactId>selenium-java</artifactId>
22       <version>4.25.0</version>
23     </dependency>
24   </dependencies>
25
26 </project>
```

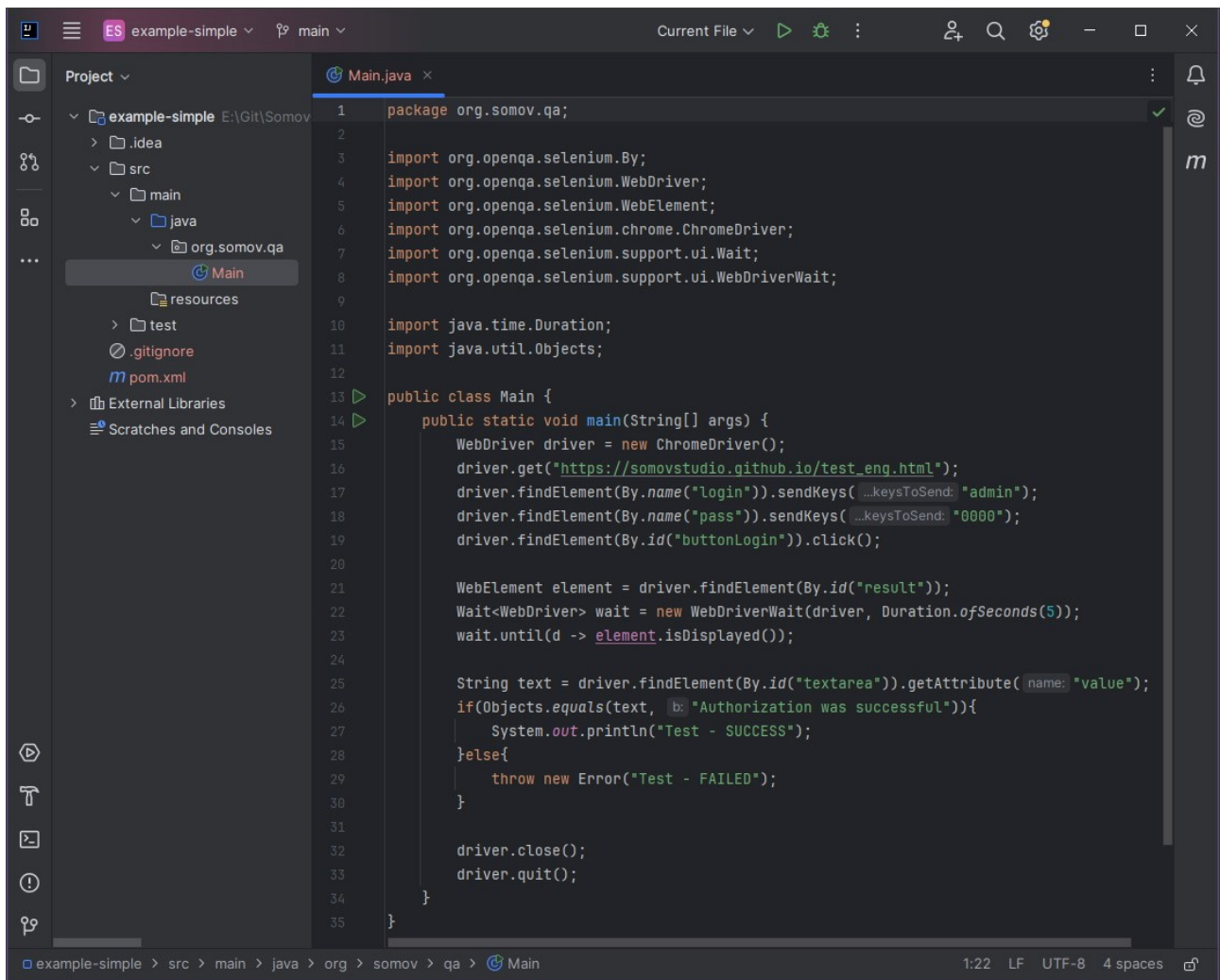
6. В файле Main.java описать автотест следующим образом

```
WebDriver driver = new ChromeDriver();
driver.get("https://somovstudio.github.io/test_eng.html");
driver.findElement(By.name("login")).sendKeys("admin");
driver.findElement(By.name("pass")).sendKeys("0000");
driver.findElement(By.id("buttonLogin")).click();

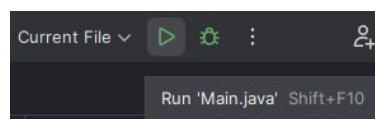
WebElement element = driver.findElement(By.id("result"));
Wait<WebDriver> wait = new WebDriverWait(driver, Duration.ofSeconds(5));
wait.until(d -> element.isDisplayed());

String text = driver.findElement(By.id("textarea")).getAttribute("value");
if(Objects.equals(text, "Authorization was successful")){
    System.out.println("Test - SUCCESS");
}else{
    throw new Error("Test - FAILED");
}

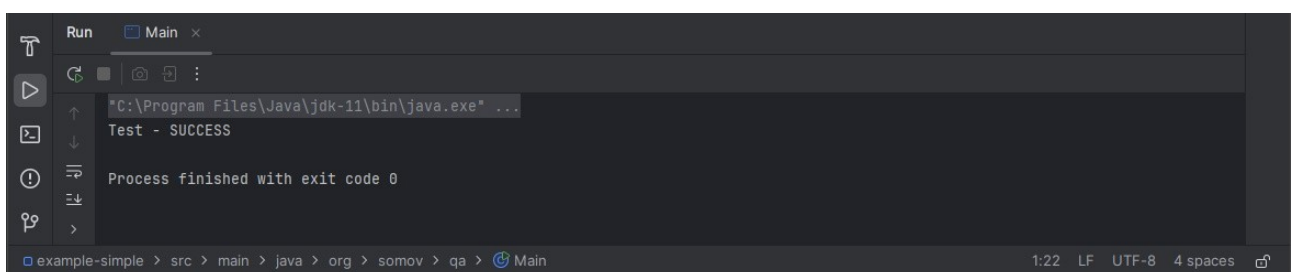
driver.close();
driver.quit();
```



7. Запустить автотест нажав кнопку Run в IntelliJ IDEA



8. результат автотеста в консоли IntelliJ IDEA



Полезные ссылки:

- Официальная документация The Selenium Browser Automation Project
<https://www.selenium.dev/documentation/>
- Документация Getting started
https://www.selenium.dev/documentation/webdriver/getting_started/
- Документация Install a Selenium library
https://www.selenium.dev/documentation/webdriver/getting_started/install_library/
- Документация API Docs
<https://www.selenium.dev/selenium/docs/api/java/index.html>
- Официальная страница IntelliJ IDEA Community Edition
<https://www.jetbrains.com/idea/download/other.html>
- Официальная страница Maven
<https://mvnrepository.com/>
- Официальная страница junit5
<https://junit.org/junit5/>
- Официальная страница TestNG
<https://testng.org/>
- Скачать Java SE Development Kit 11
<https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html>

Практика применения JUnit

1. В редакторе IntelliJ IDEA создаем новый проект example-junit

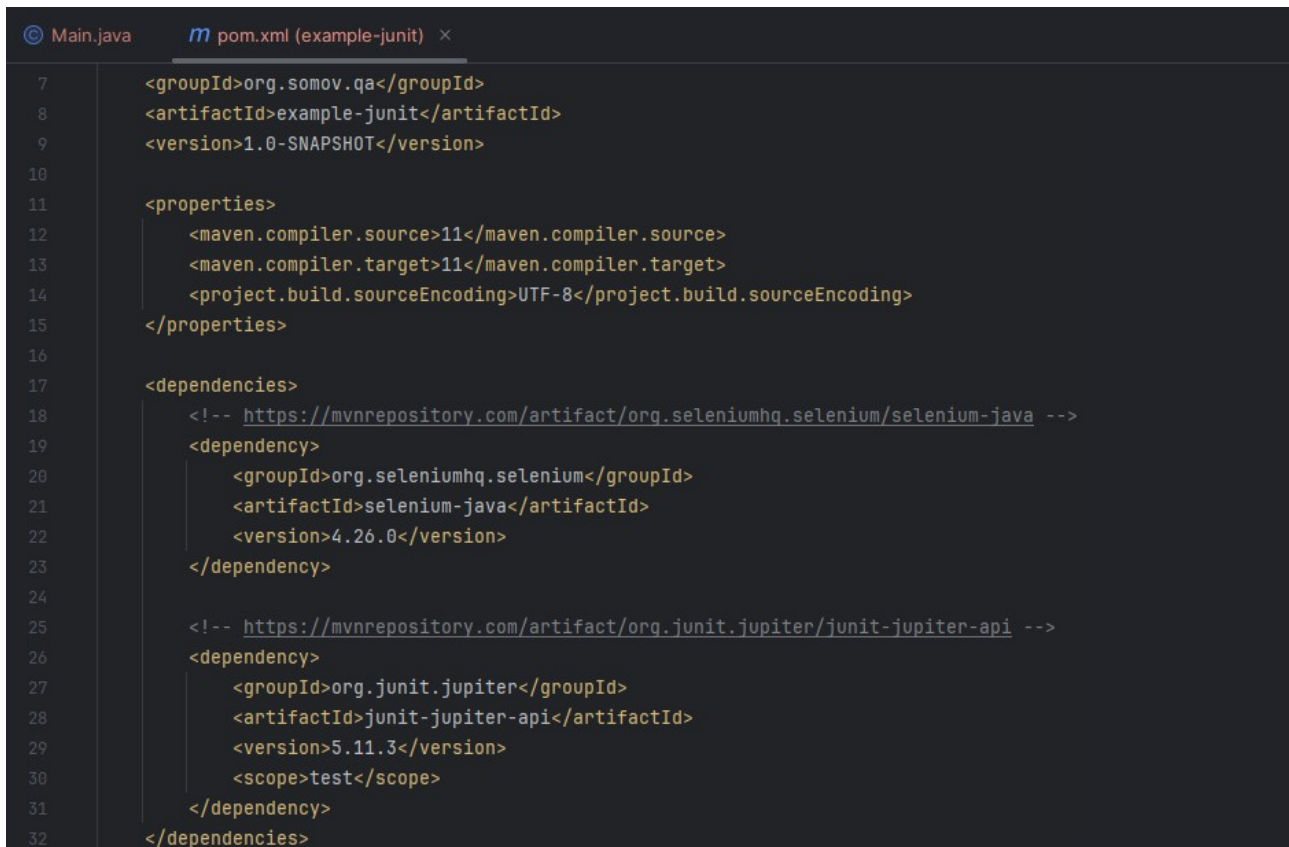
2. Подключить библиотеку Selenium и JUnit Jupiter API к проекту в файле pom.xml

ссылка: <https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>

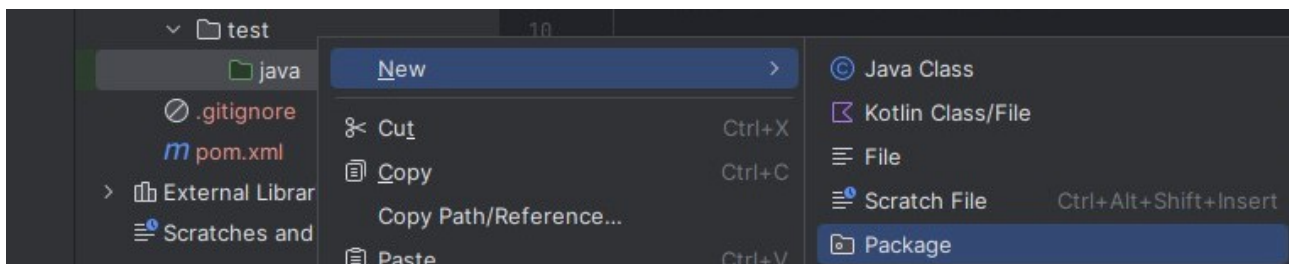
ссылка: <https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api>

```
<dependencies>
  <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.26.0</version>
  </dependency>

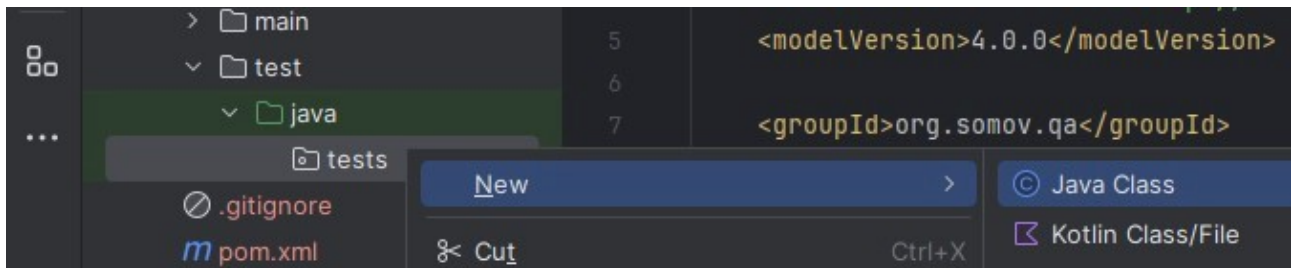
  <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.11.3</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```



3. В папке test создать пакет tests



4. В пакете tests создать класс автотеста TestAuthorization.java



5. Аннотации JUnit 5

JUnit 5 предлагает следующие аннотации для написания тестов.

Аннотации	Описание
@BeforeEach	Аннотированный метод будет запускаться перед каждым тестовым методом в тестовом классе.
@AfterEach	Аннотированный метод будет запускаться после каждого тестового метода в тестовом классе.
@BeforeAll	Аннотированный метод будет запущен перед всеми тестовыми методами в тестовом классе. Этот метод должен быть статическим.
@AfterAll	Аннотированный метод будет запущен после всех тестовых методов в тестовом классе. Этот метод должен быть статическим.
@Test	Он используется, чтобы пометить метод как тест junit.
@DisplayName	Используется для предоставления любого настраиваемого отображаемого имени для тестового класса или тестового метода
@Disable	Он используется для отключения или игнорирования тестового класса или тестового метода из набора тестов.
@Nested	Используется для создания вложенных тестовых классов
@Tag	Пометьте методы тестирования или классы тестов тегами для обнаружения и фильтрации тестов.
@TestFactory	Отметить метод - это тестовая фабрика для динамических тестов.

6. В файле TestAuthorization.java описать автотест следующим образом

```
package tests;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Disabled;
import org.junit.jupiter.api.Tag;
import org.junit.jupiter.api.Test;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Wait;
import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration;
import java.util.Objects;
```

```

public class TestAuthorization {
    public static WebDriver driver;

    @BeforeAll
    public static void before(){
        driver = new ChromeDriver();
        driver.manage().window().maximize();
    }

    @Tag("PROD")
    @Test
    public void testAuthorization(){
        driver.get("https://somovstudio.github.io/test_eng.html");
        driver.findElement(By.name("login")).sendKeys("admin");
        driver.findElement(By.name("pass")).sendKeys("0000");
        driver.findElement(By.id("buttonLogin")).click();

        WebElement element = driver.findElement(By.id("result"));
        Wait<WebDriver> wait = new WebDriverWait(driver, Duration.ofSeconds(5));
        wait.until(d -> element.isDisplayed());

        String text = driver.findElement(By.id("textarea")).getAttribute("value");
        System.out.println(text);
        Assertions.assertEquals(text, "Authorization was successful");
    }

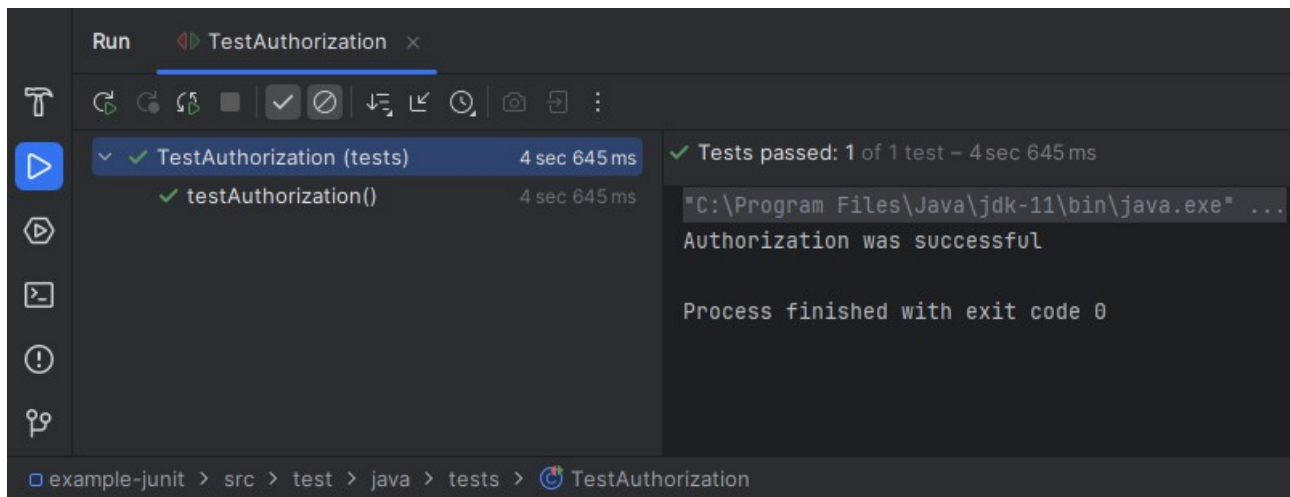
    @AfterAll
    public static void after(){
        driver.close();
        driver.quit();
    }
}

```

The screenshot shows an IDE window with the following components:

- Project Explorer (Left):** Shows a project named 'example-junit' with a structure including 'src' (main, test) and 'target' folders. The 'TestAuthorization' class is highlighted in the 'test' folder.
- Editor (Right):** Displays the code for 'TestAuthorization.java'. The code includes annotations like `@BeforeAll`, `@Tag("PROD")`, `@Test`, and `@AfterAll`. It uses `WebDriver` to interact with a web application, performing actions like `sendKeys` and `click`, and using `WebDriverWait` to wait for elements to be displayed. The code also includes assertions to verify the test results.

7. Запустить автотеста в среде IntelliJ IDEA и убедиться что всё работает корректно

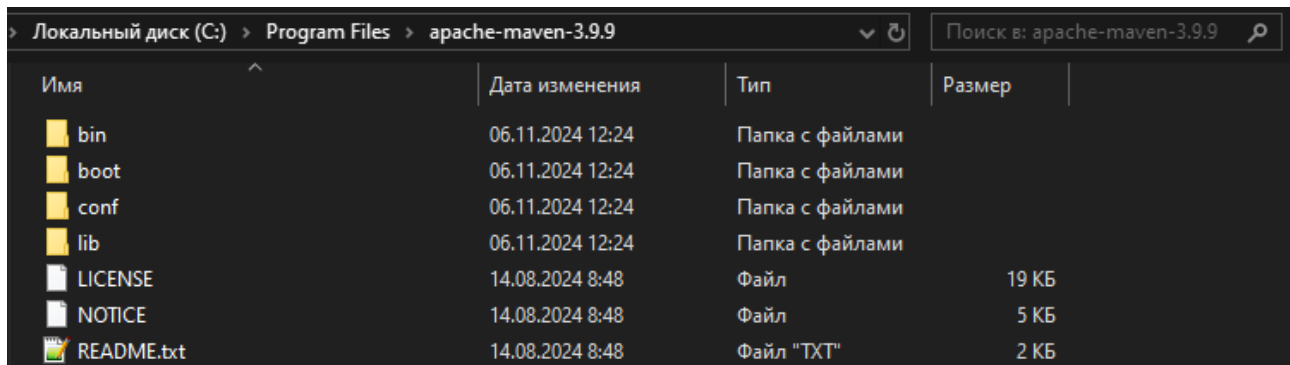


Полезные ссылки:

- Тьюториал по JUnit 5 - Введение
<https://habr.com/ru/articles/590607/>
- JUnit 5 User Guide
<https://junit.org/junit5/docs/current/user-guide/>
- How to run JUnit5 tests through Command Line
<https://qaautomation.expert/2022/05/12/how-to-run-junit5-tests-through-command-line/>
- Using JUnit 5 Platform
<https://maven.apache.org/surefire/maven-surefire-plugin/examples/junit-platform.html>
- JUnit Jupiter API
<https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api>
- JUnit Jupiter Engine
<https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-engine>
- Maven Surefire Plugin
<https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-surefire-plugin>
- Selenium Java
<https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>
- Apache Maven
<https://maven.apache.org/download.cgi>

Запуск автотестов с помощью Maven

1. Скачать и установить [Apache Maven](#) (архив: apache-maven-3.9.9-bin.zip)



Имя	Дата изменения	Тип	Размер
bin	06.11.2024 12:24	Папка с файлами	
boot	06.11.2024 12:24	Папка с файлами	
conf	06.11.2024 12:24	Папка с файлами	
lib	06.11.2024 12:24	Папка с файлами	
LICENSE	14.08.2024 8:48	Файл	19 КБ
NOTICE	14.08.2024 8:48	Файл	5 КБ
README.txt	14.08.2024 8:48	Файл "TXT"	2 КБ

2. Подключить плагин Maven Surefire Plugin к проекту в файле pom.xml

ссылка: <https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-surefire-plugin>

ссылка: <https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-engine>

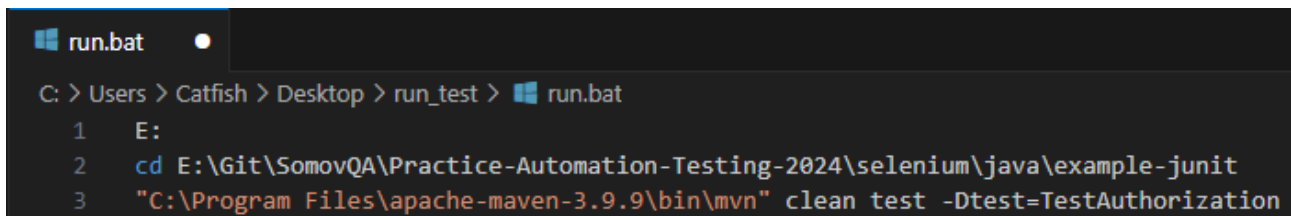
```
<properties>
  <maven.compiler.source>11</maven.compiler.source>
  <maven.compiler.target>11</maven.compiler.target>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
  <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.26.0</version>
  </dependency>

  <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.11.3</version>
    <scope>test</scope>
  </dependency>

  <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-engine -->
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>5.11.3</version>
    <scope>test</scope>
  </dependency>
</dependencies>
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>3.5.2</version>
      <dependencies>
        <dependency>
          <groupId>org.junit.jupiter</groupId>
          <artifactId>junit-jupiter-engine</artifactId>
          <version>5.11.3</version>
        </dependency>
      </dependencies>
    </plugin>
  </plugins>
</build>
```


3. Создать run.bat файл для запуска автотеста из командной строки с помощью Maven

```
E:
cd E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\java\example-junit
"C:\Program Files\apache-maven-3.9.9\bin\mvn" clean test -Dtest=TestAuthorization
```

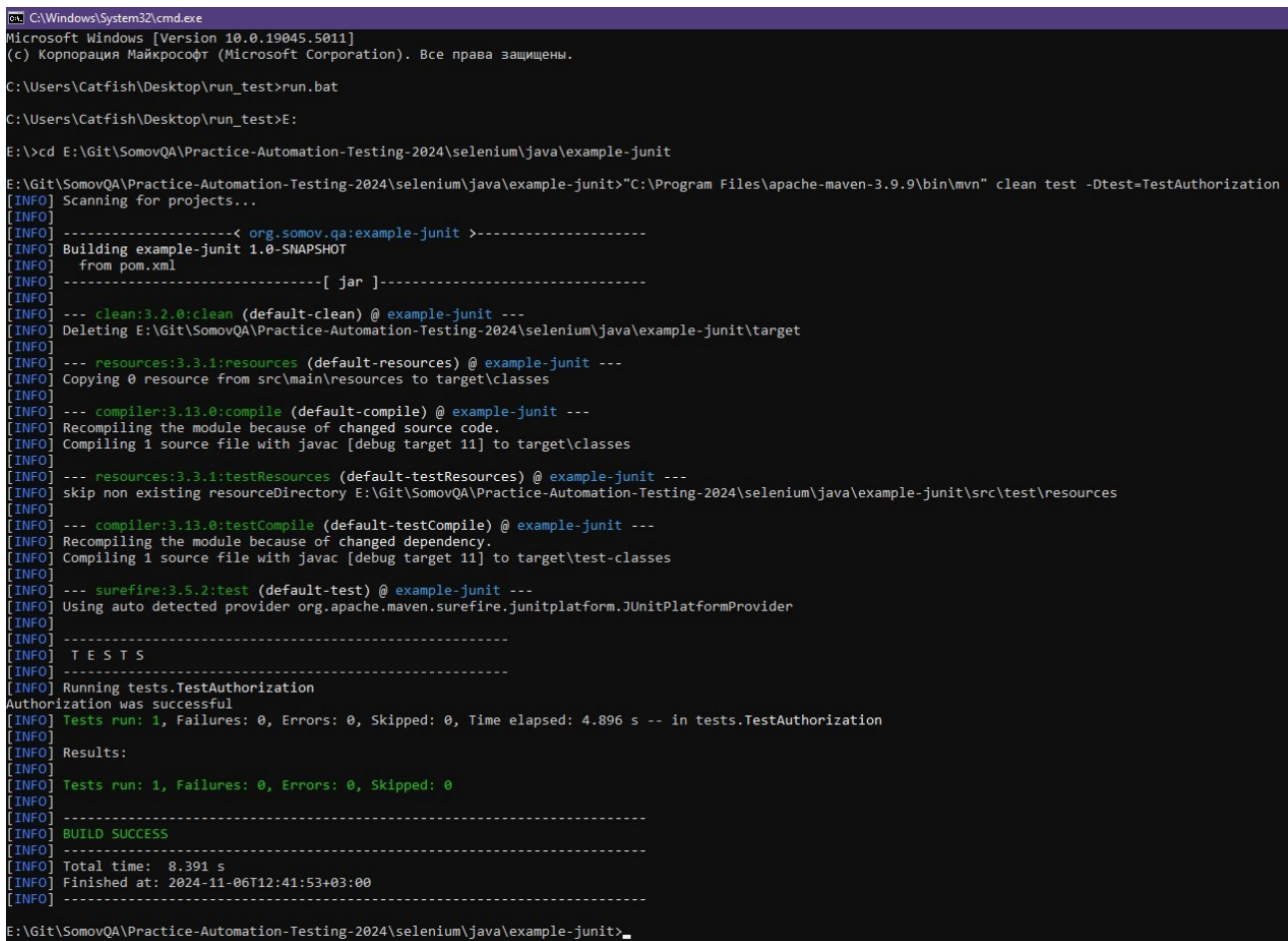


```
run.bat

C: > Users > Catfish > Desktop > run_test > run.bat

1 E:
2 cd E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\java\example-junit
3 "C:\Program Files\apache-maven-3.9.9\bin\mvn" clean test -Dtest=TestAuthorization
```

Запустить файл run.bat



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.5011]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Catfish\Desktop\run_test>run.bat
C:\Users\Catfish\Desktop\run_test>E:
E:\>cd E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\java\example-junit
E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\java\example-junit>"C:\Program Files\apache-maven-3.9.9\bin\mvn" clean test -Dtest=TestAuthorization
[INFO] Scanning for projects...
[INFO]
[INFO] -----< org.somov.qa:example-junit >-----
[INFO] Building example-junit 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- clean:3.2.0:clean (default-clean) @ example-junit ---
[INFO] Deleting E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\java\example-junit\target
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ example-junit ---
[INFO] Copying 0 resource from src/main/resources to target\classes
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ example-junit ---
[INFO] Recompiling the module because of changed source code.
[INFO] Compiling 1 source file with javac [debug target 11] to target\classes
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ example-junit ---
[INFO] skip non existing resourceDirectory E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\java\example-junit\src\test\resources
[INFO]
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ example-junit ---
[INFO] Recompiling the module because of changed dependency.
[INFO] Compiling 1 source file with javac [debug target 11] to target\test-classes
[INFO]
[INFO] --- surefire:3.5.2:test (default-test) @ example-junit ---
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running tests.TestAuthorization
Authorization was successful
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 4.896 s -- in tests.TestAuthorization
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 8.391 s
[INFO] Finished at: 2024-11-06T12:41:53+03:00
[INFO]
E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\java\example-junit>
```

Практика применения TestNG

1. В редакторе IntelliJ IDEA создаем новый проект example-testng

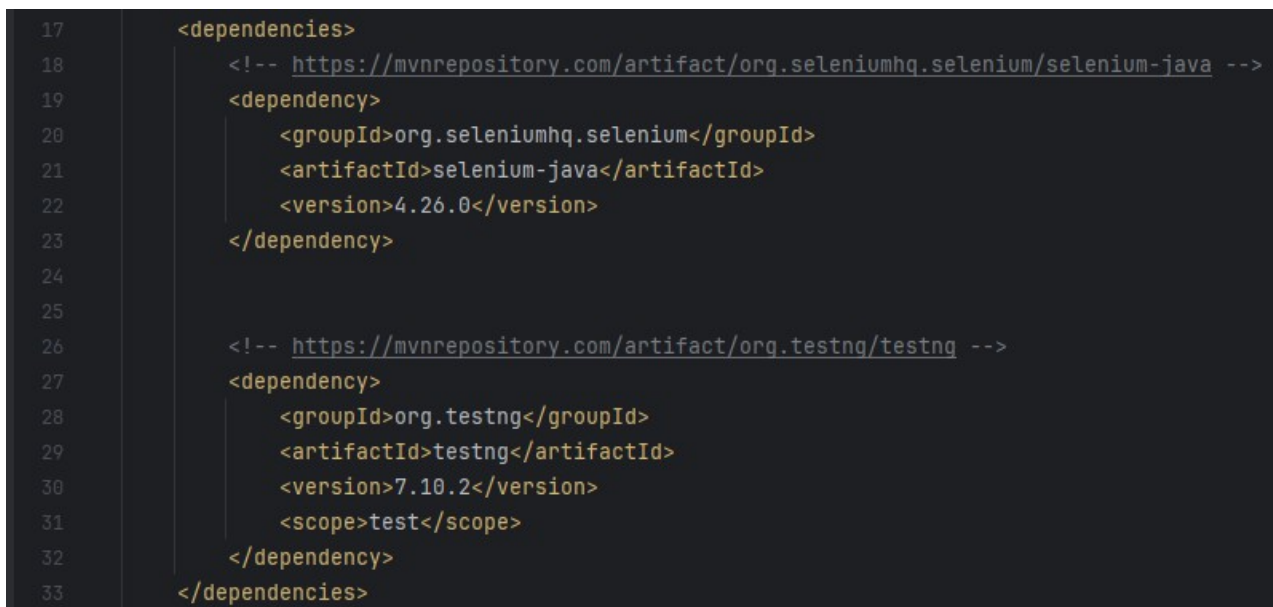
2. Подключить библиотеку Selenium и TestNG к проекту в файле pom.xml

ссылка: <https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>

ссылка: <https://mvnrepository.com/artifact/org.testng/testng>

```
<dependencies>
  <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.26.0</version>
  </dependency>

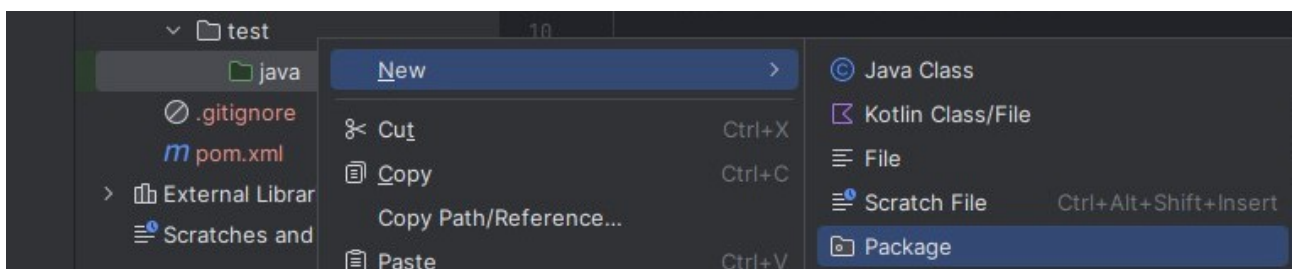
  <!-- https://mvnrepository.com/artifact/org.testng/testng -->
  <dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>7.10.2</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```



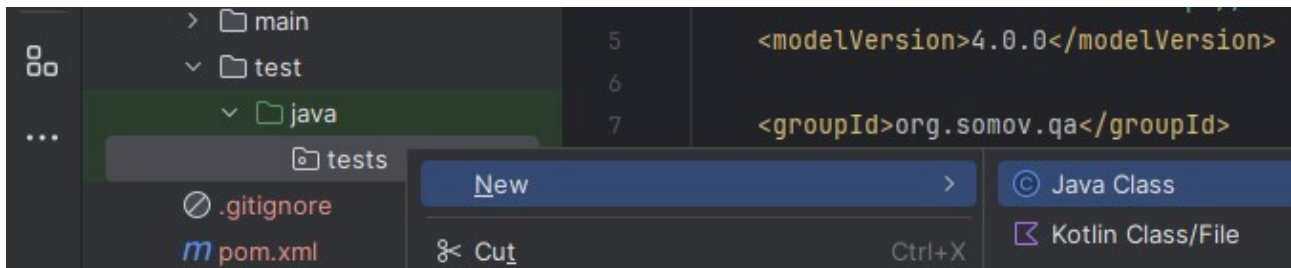
The screenshot shows the IntelliJ IDEA IDE with a dark theme. On the left, a file explorer shows a project structure with a 'test' directory containing a 'java' subdirectory. The main editor area displays the 'pom.xml' file. The XML content is as follows:

```
17 <dependencies>
18   <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
19   <dependency>
20     <groupId>org.seleniumhq.selenium</groupId>
21     <artifactId>selenium-java</artifactId>
22     <version>4.26.0</version>
23   </dependency>
24
25   <!-- https://mvnrepository.com/artifact/org.testng/testng -->
26   <dependency>
27     <groupId>org.testng</groupId>
28     <artifactId>testng</artifactId>
29     <version>7.10.2</version>
30     <scope>test</scope>
31   </dependency>
32 </dependencies>
33
```

3. В папке test создать пакет tests



4. В пакете tests создать класс автотеста TestAuthorization.java



5. В файле TestAuthorization.java описать автотест следующим образом

```
package tests;

import org.testng.Assert;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Wait;
import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration;
import java.util.Objects;

public class TestAuthorization {
    public static WebDriver driver;

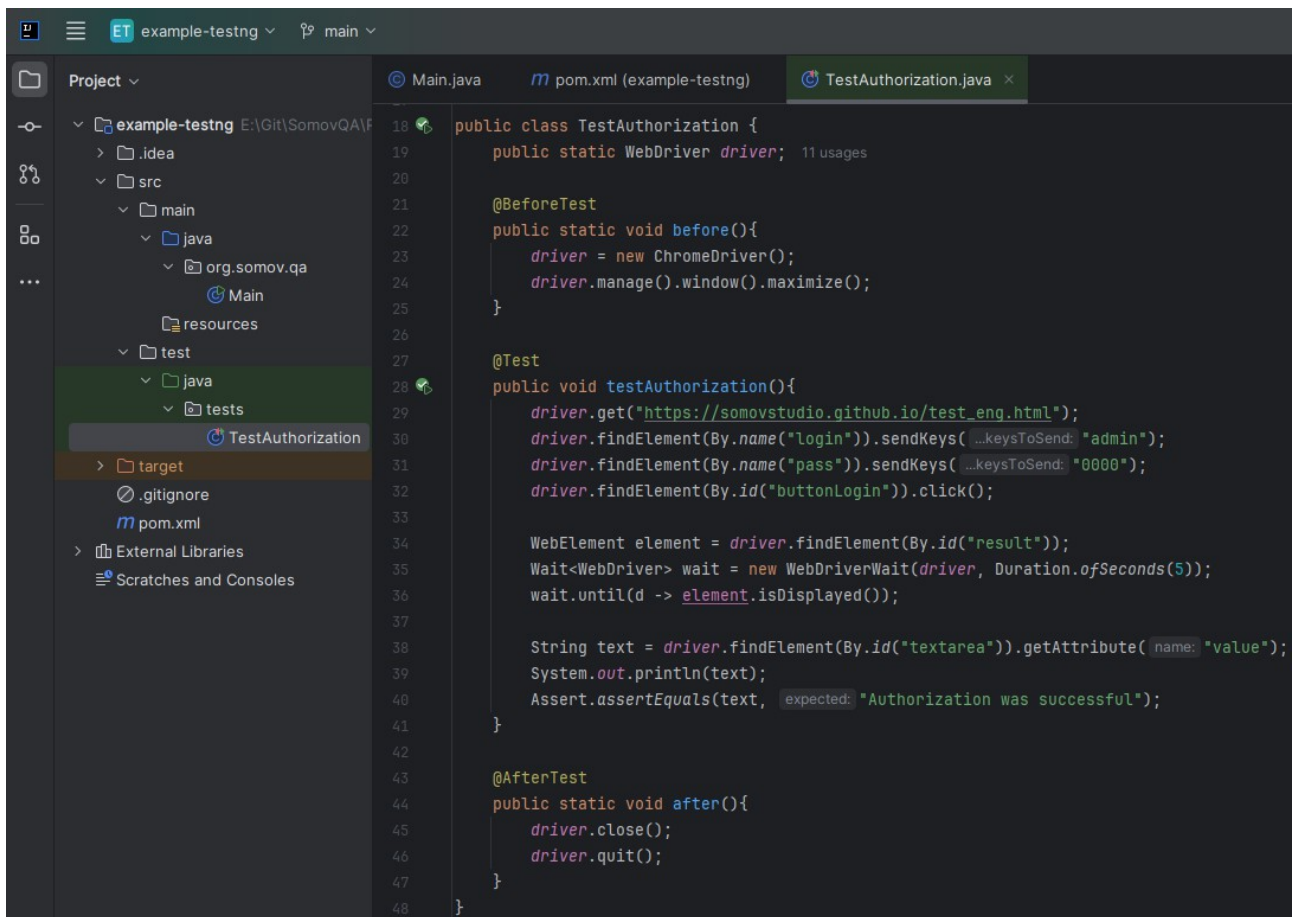
    @BeforeTest
    public static void before() {
        driver = new ChromeDriver();
        driver.manage().window().maximize();
    }

    @Test
    public void testAuthorization() {
        driver.get("https://somovstudio.github.io/test_eng.html");
        driver.findElement(By.name("login")).sendKeys("admin");
        driver.findElement(By.name("pass")).sendKeys("0000");
        driver.findElement(By.id("buttonLogin")).click();

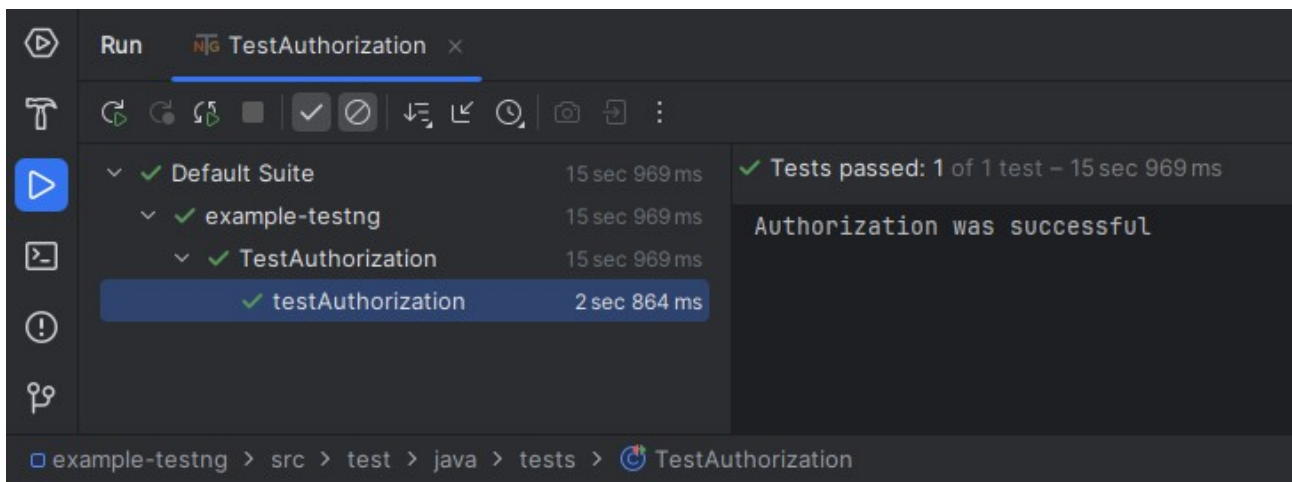
        WebElement element = driver.findElement(By.id("result"));
        Wait<WebDriver> wait = new WebDriverWait(driver, Duration.ofSeconds(5));
        wait.until(d -> element.isDisplayed());

        String text = driver.findElement(By.id("textarea")).getAttribute("value");
        System.out.println(text);
        Assert.assertEquals(text, "Authorization was successful");
    }

    @AfterTest
    public static void after() {
        driver.close();
        driver.quit();
    }
}
```



7. Запустить автотеста в среде IntelliJ IDEA и убедиться что всё работает корректно



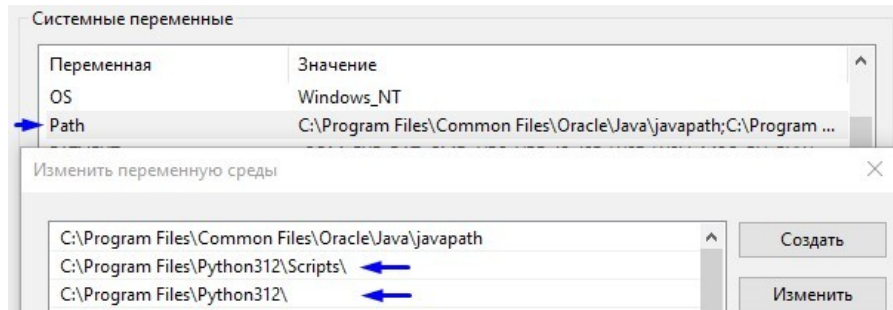
Полезные ссылки:

- Selenium Java
<https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>
- TestNG
<https://mvnrepository.com/artifact/org.testng/testng>

Создание проекта Selenium (Python) с простым автотестом

1. Скачать и установить [Python](#)

2. В переменной Path прописать путь к папке C:\Program Files\Python
(Панель управления > Система > Дополнительные параметры системы > Переменные среды)



3. Проверить версию Python, посмотреть установленные библиотеки, обновить pip

```
python -V
pip list
python -m pip install --upgrade pip
```

```
Командная строка
Microsoft Windows [Version 10.0.19045.4894]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Catfish>python -V
Python 3.12.6

C:\Users\Catfish>pip list
Package Version
-----
pip      24.2

C:\Users\Catfish>python -m pip install --upgrade pip
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pip in c:\program files\python312\lib\site-packages (24.2)

C:\Users\Catfish>
```

4. Установить Selenium командой

```
pip install -U selenium
```

```
Командная строка
Microsoft Windows [Version 10.0.19045.4894]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Catfish>pip install -U selenium
Defaulting to user installation because normal site-packages is not writeable
Collecting selenium
  Downloading selenium-4.24.0-py3-none-any.whl.metadata (7.1 kB)
Collecting urllib3<3,>=1.26 (from urllib3[socks]<3,>=1.26->selenium)
  Downloading urllib3-2.2.3-py3-none-any.whl.metadata (6.5 kB)
```

5. Проверить установленную версию Selenium

```
pip list
```



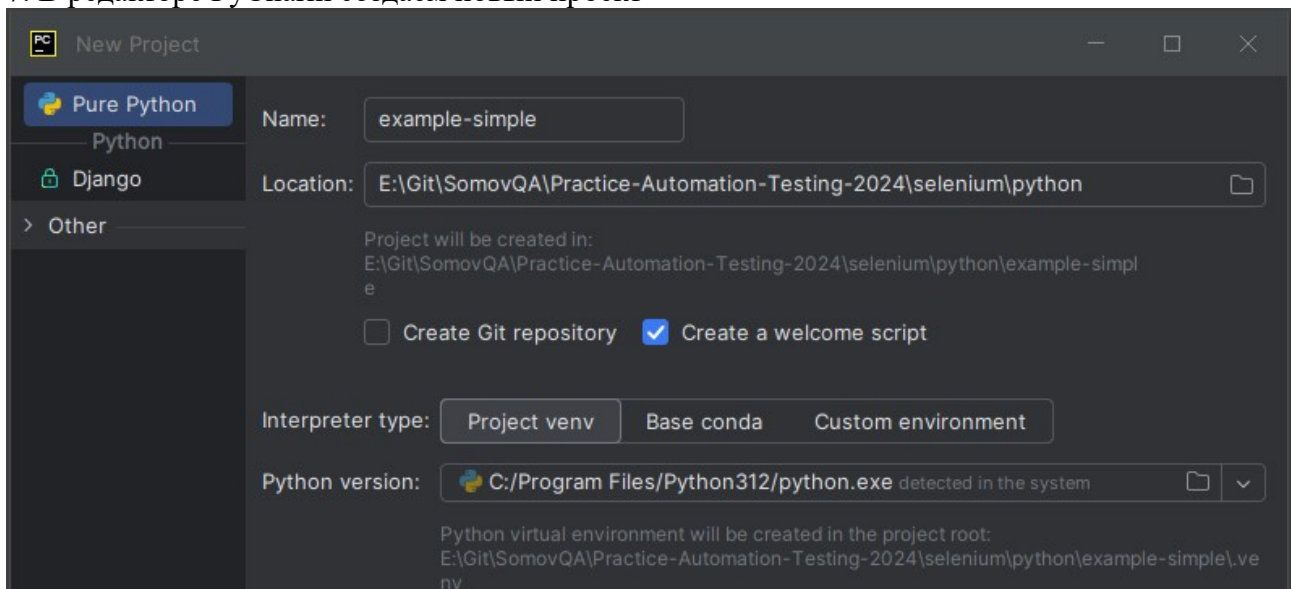
```
Командная строка
Microsoft Windows [Version 10.0.19045.4894]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Catfish>pip list
Package            Version
-----
attrs              24.2.0
certifi            2024.8.30
cffi               1.17.1
h11                0.14.0
idna               3.10
outcome            1.3.0.post0
pip               24.2
pyparser           2.22
PySocks            1.7.1
selenium           4.24.0
sniffio            1.3.1
sortedcontainers   2.4.0
trio               0.26.2
trio-websocket     0.11.1
typing_extensions 4.12.2
urllib3            2.2.3
websocket-client   1.8.0
wsproto            1.2.0

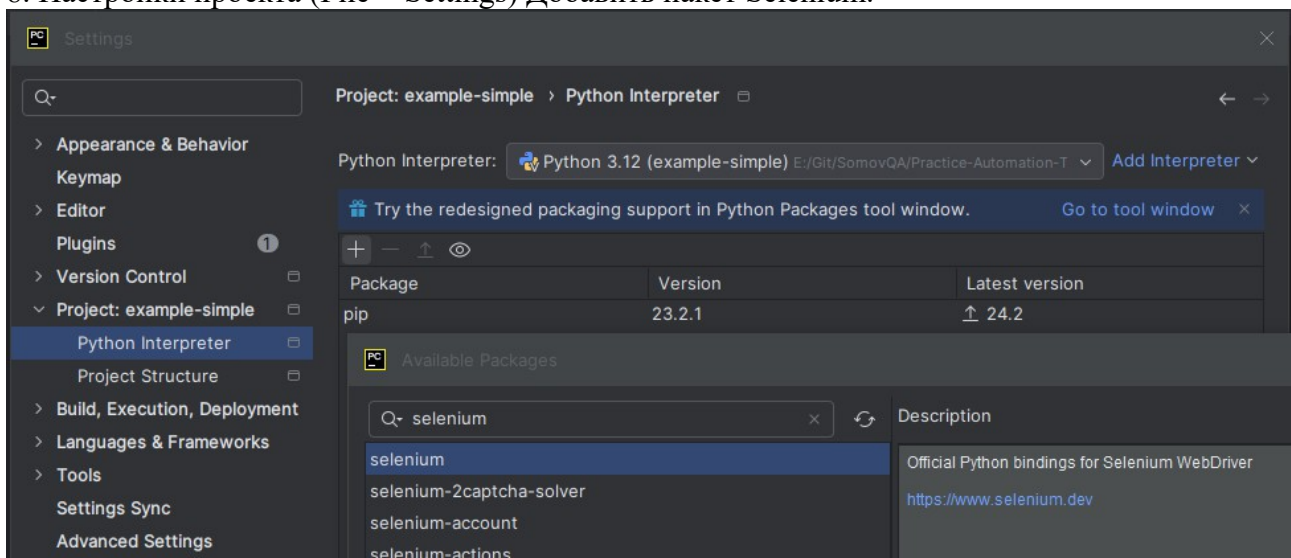
C:\Users\Catfish>
```

6. Скачать и установить [PyCharm Community Edition](#)

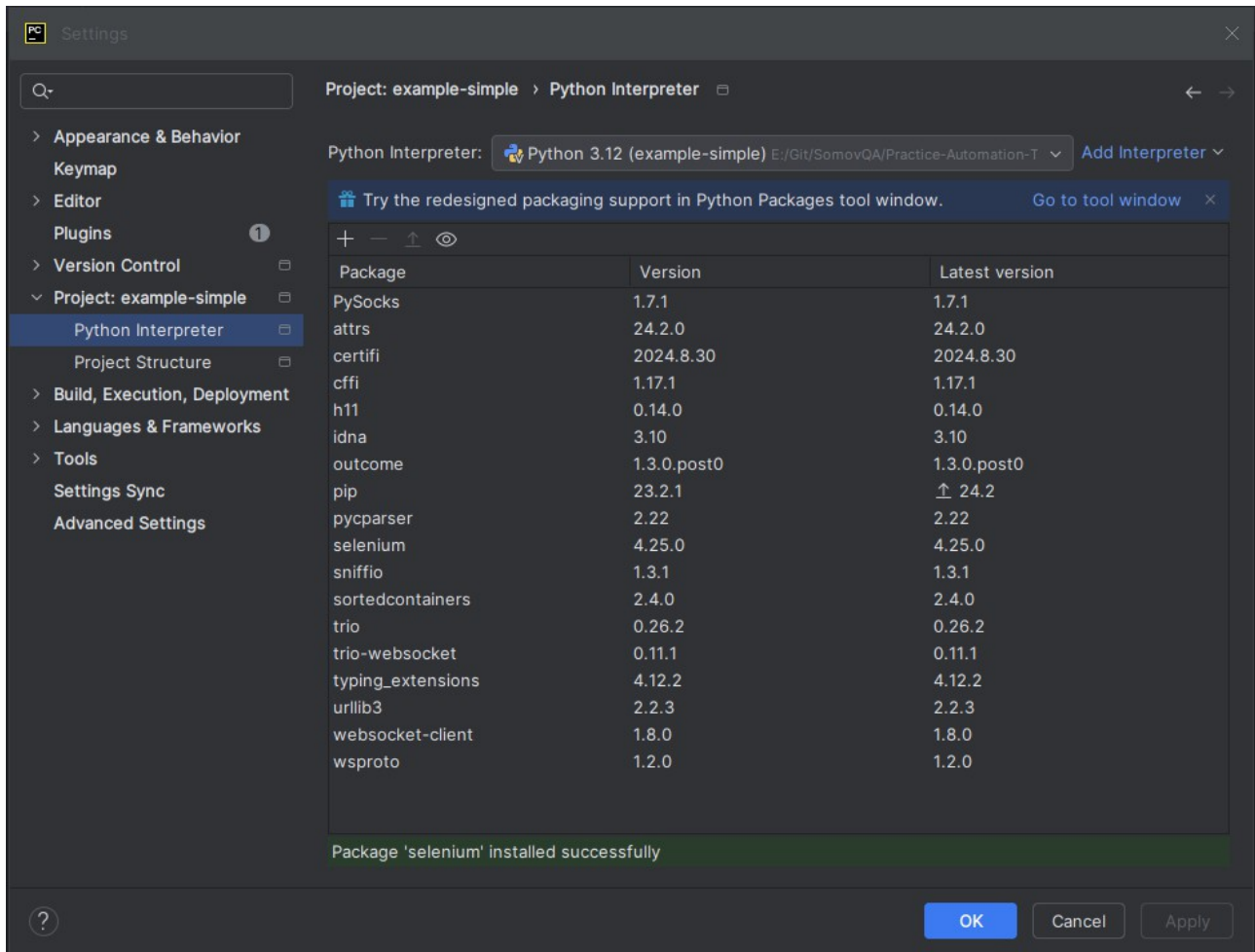
7. В редакторе PyCharm создаем новый проект



8. Настройки проекта (File > Settings) Добавить пакет Selenium.



При установке будут добавлены следующие пакеты

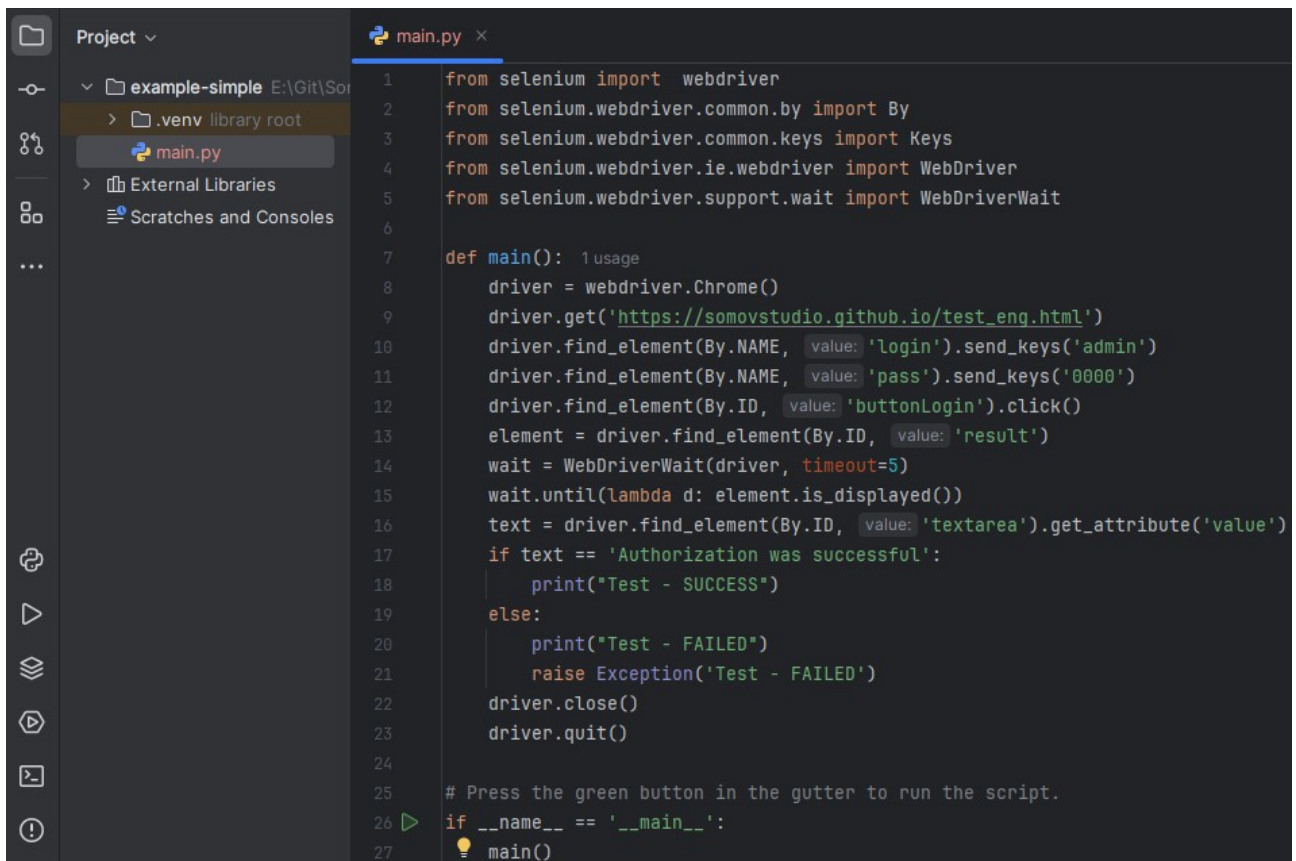


9. В файле main.py описать автотест следующим образом

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.ie.webdriver import WebDriver
from selenium.webdriver.support.wait import WebDriverWait

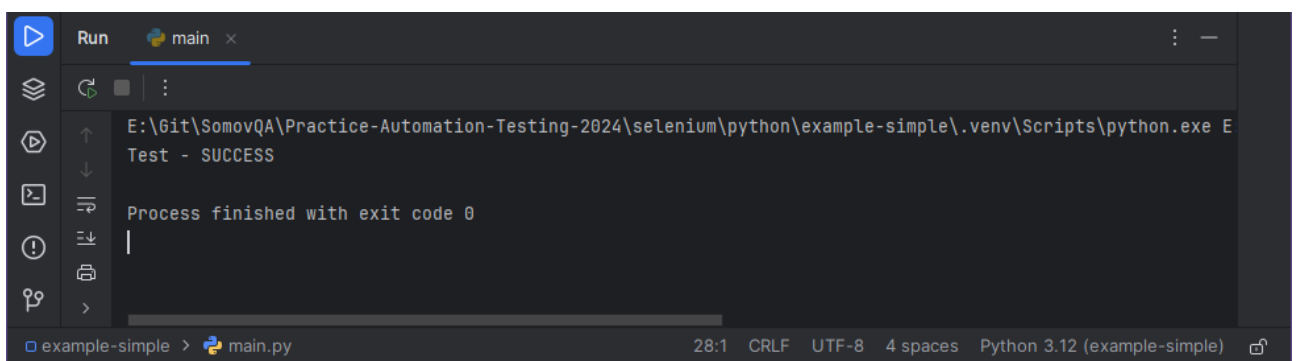
def main():
    driver = webdriver.Chrome()
    driver.get('https://somovstudio.github.io/test_eng.html')
    driver.find_element(By.NAME, 'login').send_keys('admin')
    driver.find_element(By.NAME, 'pass').send_keys('0000')
    driver.find_element(By.ID, 'buttonLogin').click()
    element = driver.find_element(By.ID, 'result')
    wait = WebDriverWait(driver, timeout=5)
    wait.until(lambda d: element.is_displayed())
    text = driver.find_element(By.ID, 'textarea').get_attribute('value')
    if text == 'Authorization was successful':
        print("Test - SUCCESS")
    else:
        print("Test - FAILED")
        raise Exception('Test - FAILED')
    driver.close()
    driver.quit()

# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    main()
```



```
1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 from selenium.webdriver.common.keys import Keys
4 from selenium.webdriver.ie.webdriver import WebDriver
5 from selenium.webdriver.support.wait import WebDriverWait
6
7 def main():
8     driver = webdriver.Chrome()
9     driver.get('https://somovstudio.github.io/test_eng.html')
10    driver.find_element(By.NAME, 'login').send_keys('admin')
11    driver.find_element(By.NAME, 'pass').send_keys('0000')
12    driver.find_element(By.ID, 'buttonLogin').click()
13    element = driver.find_element(By.ID, 'result')
14    wait = WebDriverWait(driver, timeout=5)
15    wait.until(lambda d: element.is_displayed())
16    text = driver.find_element(By.ID, 'textarea').get_attribute('value')
17    if text == 'Authorization was successful':
18        print("Test - SUCCESS")
19    else:
20        print("Test - FAILED")
21        raise Exception('Test - FAILED')
22    driver.close()
23    driver.quit()
24
25 # Press the green button in the gutter to run the script.
26 if __name__ == '__main__':
27     main()
```

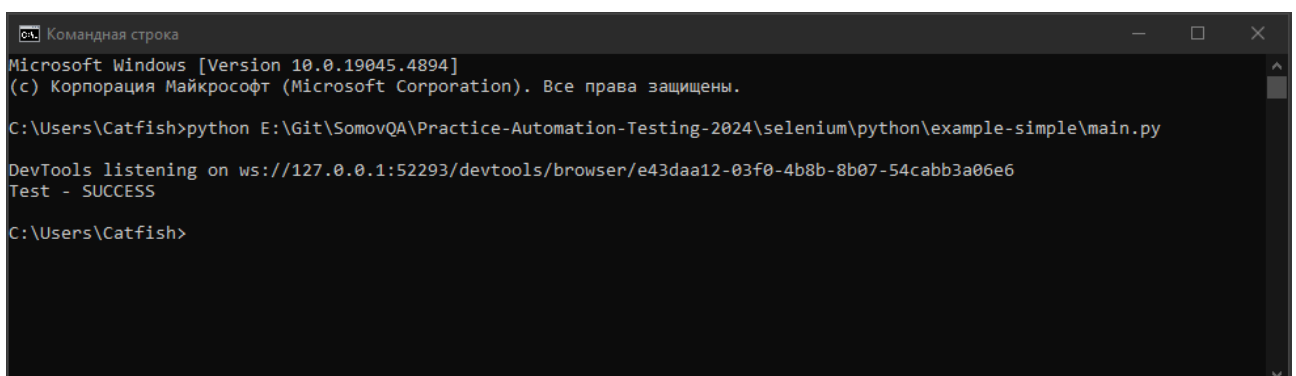
10. Запуск автотеста в редакторе PyCharm



```
Run main
E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-simple\.venv\Scripts\python.exe E
Test - SUCCESS
Process finished with exit code 0
```

11. Запуск автотеста из консоли

```
python E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-simple\main.py
```



```
Командная строка
Microsoft Windows [Version 10.0.19045.4894]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Catfish>python E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-simple\main.py

DevTools listening on ws://127.0.0.1:52293/devtools/browser/e43daa12-03f0-4b8b-8b07-54cabb3a06e6
Test - SUCCESS

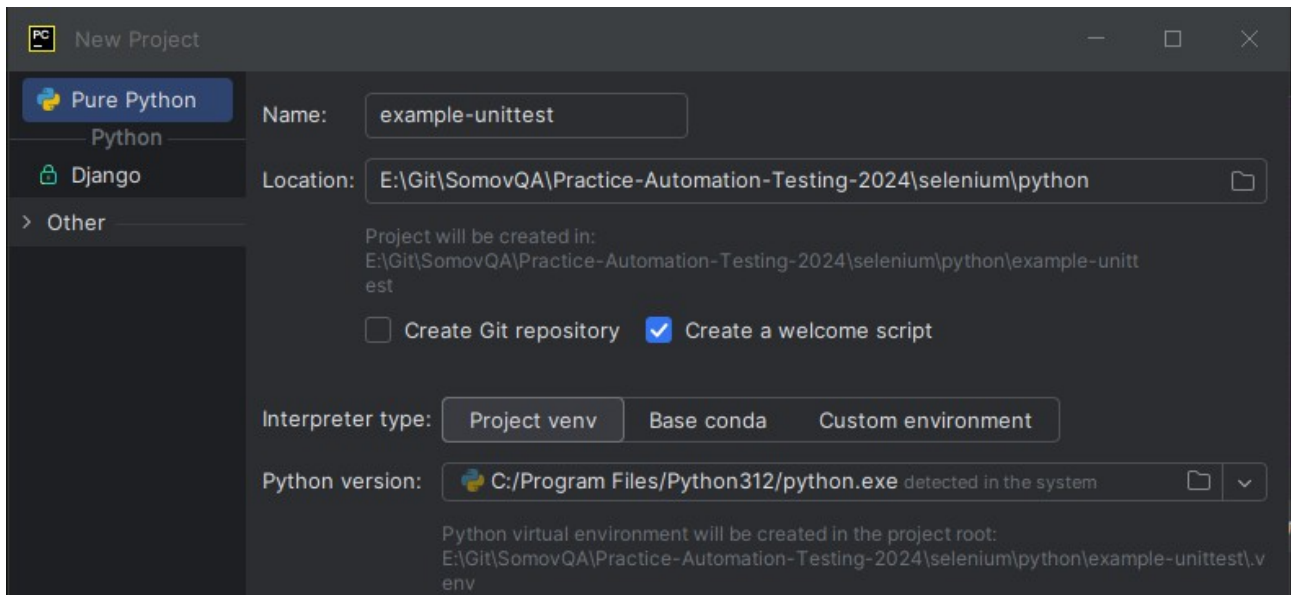
C:\Users\Catfish>
```


Полезные ссылки:

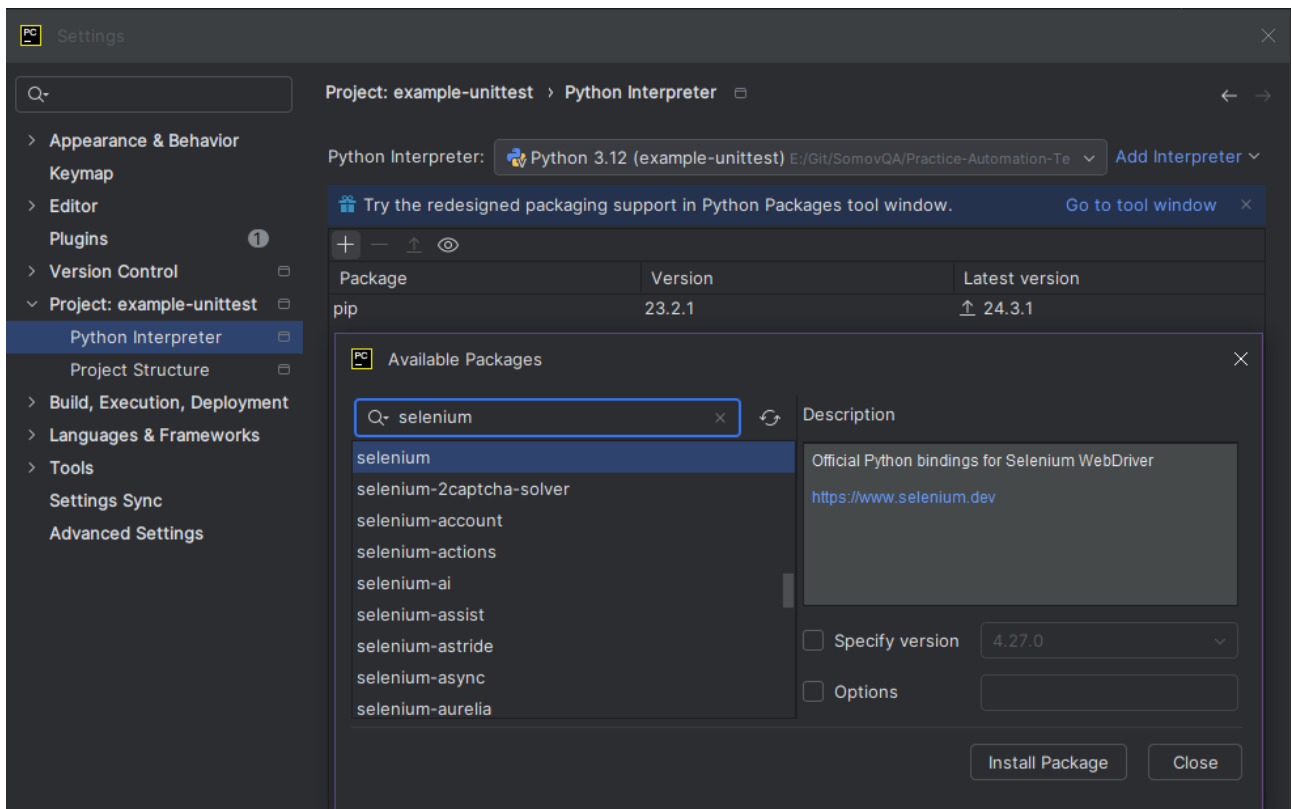
- Официальная документация The Selenium Browser Automation Project
<https://www.selenium.dev/documentation/>
- Документация API Docs (python)
<https://www.selenium.dev/selenium/docs/api/py/index.html>
- Официальная страница PyCharm Community Edition
<https://www.jetbrains.com/pycharm/download/other.html>
- Официальная страница Python
<https://www.python.org/>

Практика применения Unittest (Python)

1. В редакторе PyCharm создаем новый проект

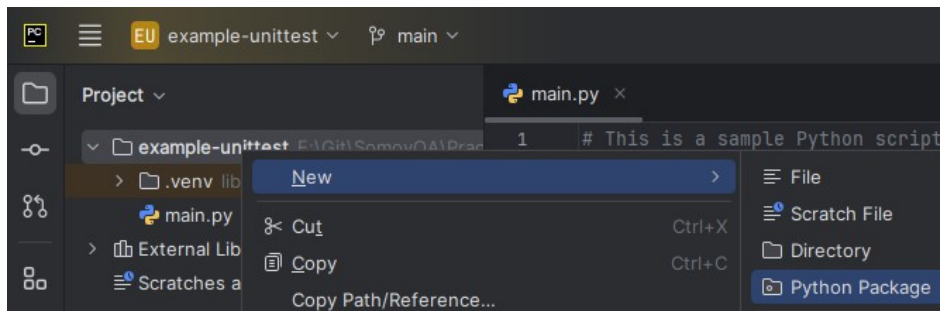


2. Настройки проекта (File > Settings) Добавить пакет Selenium.

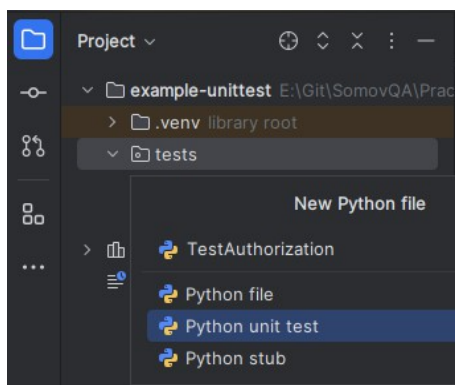


При установке будут добавлены пакеты Selenium

3. Создать пакет с именем tests



4. В пакете tests создать модульный тест (Python unit test) с именем TestAuthorization.py



5. В файле TestAuthorization.py описать автотест следующим образом

```
import unittest

from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.ie.webdriver import WebDriver
from selenium.webdriver.support.wait import WebDriverWait

class MyTestCase(unittest.TestCase):
    def test_something(self):
        driver = webdriver.Chrome()
        driver.maximize_window()
        driver.get('https://somovstudio.github.io/test_eng.html')
        driver.find_element(By.NAME, 'login').send_keys('admin')
        driver.find_element(By.NAME, 'pass').send_keys('0000')
        driver.find_element(By.ID, 'buttonLogin').click()
        element = driver.find_element(By.ID, 'result')
        wait = WebDriverWait(driver, timeout=5)
        wait.until(lambda d: element.is_displayed())
        text = driver.find_element(By.ID, 'textarea').get_property('value')
        print("Get message: " + text)

        self.assertEqual(text, 'Authorization was successful')

        driver.close()
        driver.quit()

if __name__ == '__main__':
    unittest.main()
```

6. Запуск автотеста в редакторе PyCharm и получить следующий результат:

```
Get message: Authorization was successful
Ran 1 test in 11.496s
OK
Process finished with exit code 0
```


Тестирование API с помощью Bruno

1. Скачать и установить [Wampserver](#) (сервер Apache, PHP, MySQL)
2. Создать на сервере тестовое API в папке \wamp64\www\api файл: auth.php

```
<?php

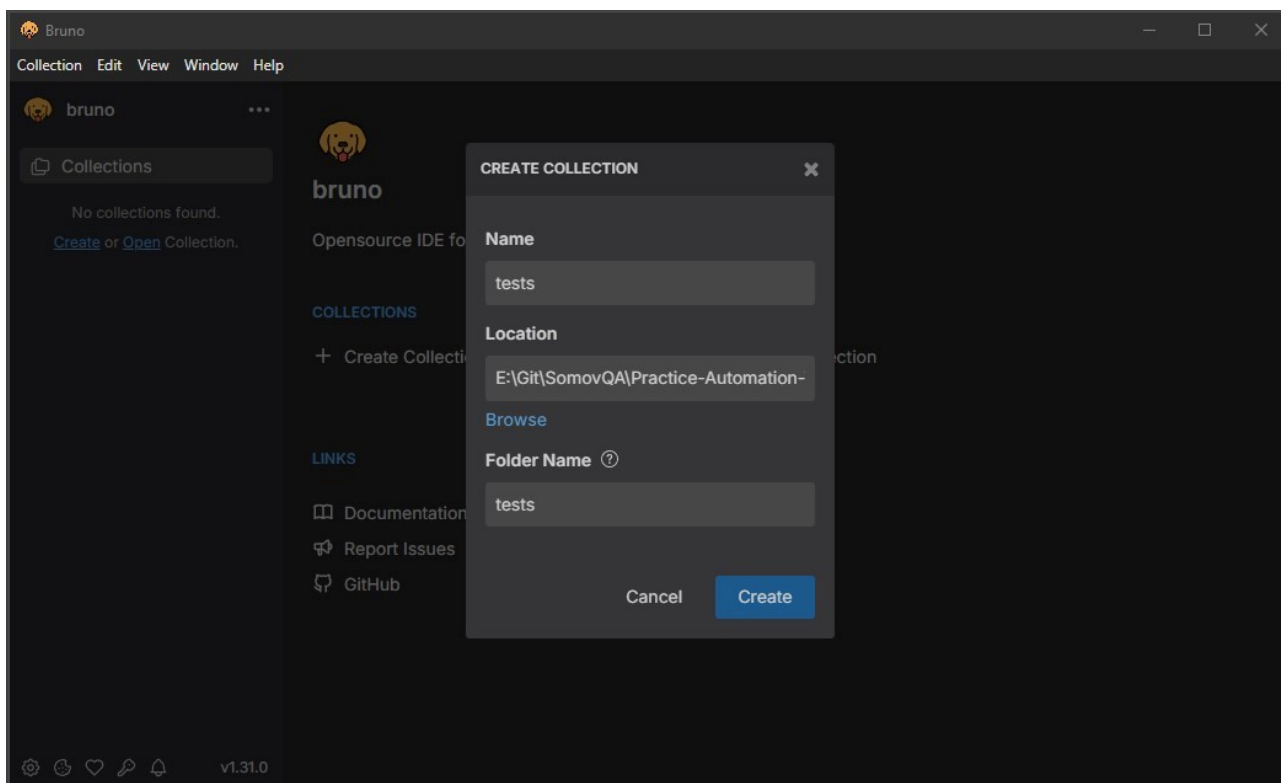
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
date_default_timezone_set("Europe/Moscow");

/*
    Пример обращения к этому API
    http://localhost/api/auth.php?name=admin&pass=0000
*/

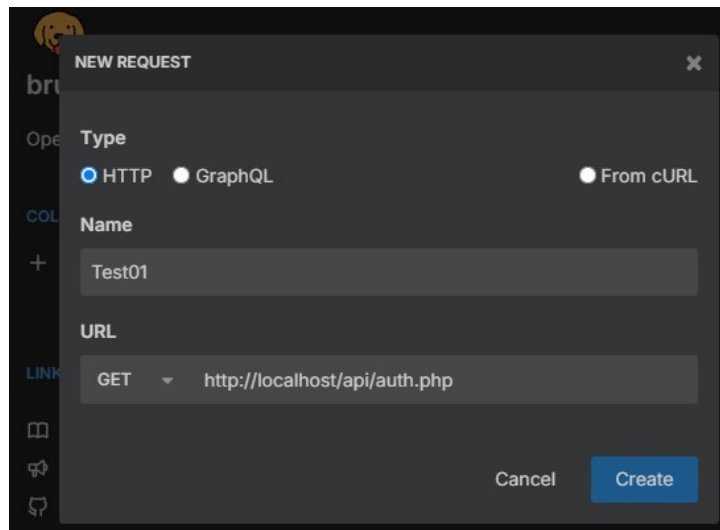
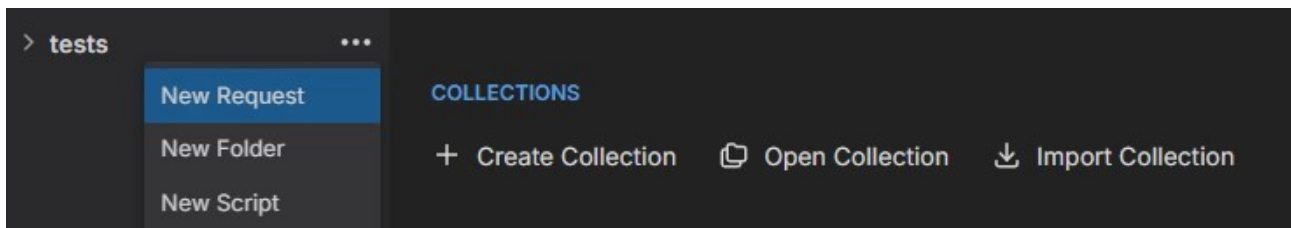
if (isset($_GET["name"]) && isset($_GET["pass"]))
{
    if ($_GET["name"] == "admin" && $_GET["pass"] == "0000")
        print('{"status":"PASSED","message":"Авторизация прошла успешно"}');
    else
        print('{"status":"FAILED","message":"Некорректный логин или пароль"}');
}
elseif (isset($_POST["post_name"]) && isset($_POST["post_pass"]))
{
    if ($_POST["post_name"] == "admin" && $_POST["post_pass"] == "0000")
        print('{"status":"PASSED","message":"Авторизация прошла успешно"}');
    else
        print('{"status":"FAILED","message":"Некорректный логин или пароль"}');
}
else
{
    print('{"status":"ERROR","message":"Нет данных для авторизации"}');
}
?>
```

таким образом API будет по адресу <http://localhost/api/auth.php?name=admin&pass=0000>

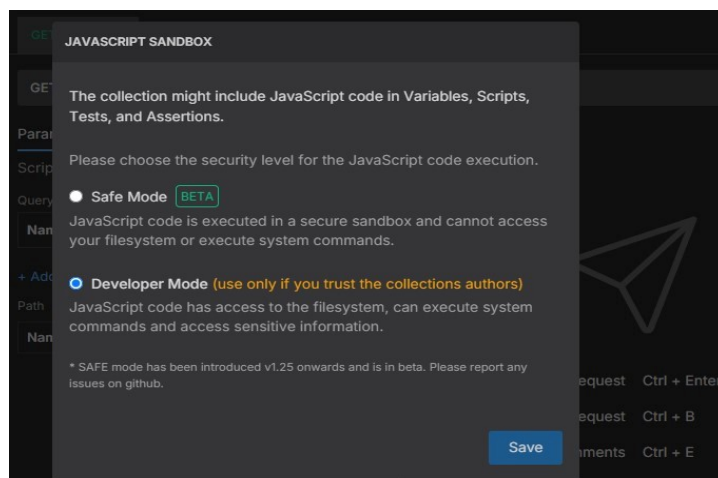
3. Скачать и установить [Bruno](#)
4. Запустить Bruno и создать коллекцию tests



5. В коллекции tests создать запрос Test01



Выбрать один из режимов (safe - ограниченный, Developer - полный)



6. В новом запросе указать параметры

name: admin

pass: 0000

Params ²		Body [*]	Headers ¹	Auth	Vars	Script	Assert	Tests	Docs
Query									
Name	Path								
name	admin							<input checked="" type="checkbox"/>	
pass	0000							<input checked="" type="checkbox"/>	

и заголовок

Content-type: application/json; charset=UTF-8

Params ² Body * Headers ¹ Auth Vars Script Assert Tests Docs		
Name	Value	
Content-type	application/json; charset=UTF-8	<input checked="" type="checkbox"/>

нажать кнопку сохранить



7. Выполните запрос

В результате сервер должен ответить: `{"status":"PASSED","message":"Авторизация прошла успешно"}`

GET Test01 x +

GET http://localhost/api/auth.php?name=admin&pass=0000

Params ² Body * Headers¹ Auth Vars Script

Assert Tests Docs

Query

Name	Path	
name	admin	<input checked="" type="checkbox"/>
pass	0000	<input checked="" type="checkbox"/>

+ Add Param

Path ?

Name	Value
------	-------

Response Headers⁸ Timeline Tests

200 OK 3ms 82B

```
1 {
2   "status": "PASSED",
3   "message": "Авторизация прошла успешно"
4 }
```

Для того чтобы выполнить POST запрос параметры нужно передавать через Body

post_name: admin

post_pass: 0000

POST Test02 x +

POST http://localhost/api/auth.php

Params Body * Headers Auth Vars Script *

Assert Tests Docs

Multipart Form

Key	Value	
post_name	admin	<input checked="" type="checkbox"/>
post_pass	0000	<input checked="" type="checkbox"/>

Response Headers⁸ Timeline Tests

200 OK 3ms 82B

```
1 {
2   "status": "PASSED",
3   "message": "Авторизация прошла успешно"
4 }
```

8. Простая проверка в которой определяется ответ сервера, если 200 значит успешно.

```
test("Проверить ответ сервера", function() {  
  const data = res.getBody();  
  expect(res.getStatus()).toEqual(200);  
});
```

GET Test01 +

GET http://localhost/api/auth.php?name=admin&pass=0000

Params² Body * Headers¹ Auth Vars Script Assert

Tests * Docs

```
1 test("Проверить ответ сервера", function() {  
2   const data = res.getBody();  
3   expect(res.getStatus()).toEqual(200);  
4 }  
5
```

Response Headers⁸ Timeline Tests¹

200 OK 2ms 82B

Tests (1/1), Passed: 1, Failed: 0

✓ Проверить ответ сервера

Assertions (0/0), Passed: 0, Failed: 0

Эту проверку можно выполнить без скрипта на вкладке Assert

Params² Body * Headers¹ Auth Vars Script Assert²

Tests Docs

Expr	Operator	Value	
res.status	equals	200	✓
res.body.message	equals	Авторизация прошла успешно	✓

Response Headers⁸ Timeline Tests²

200 OK 3ms 82B

Tests (0/0), Passed: 0, Failed: 0

Assertions (2/2), Passed: 2, Failed: 0

✓ res.status: eq 200

✓ res.body.message: eq Авторизация прошла успешно

9. Выполним проверку сообщения возвращаемого сервером

Создадим POST запрос и на вкладке Script пропишем отправляемые данные

```
req.setBody({  
  "post_name": "admin",  
  "post_pass": "0000"  
});
```

Params Body * Headers Auth Vars Script * Assert Tests * Docs

Pre Request

```
1 req.setBody({  
2   "post_name": "admin",  
3   "post_pass": "0000"  
4 });
```

на вкладке Tests добавим проверку сообщения полученного от сервера

```
test("Проверить сообщения частично", function() {  
  const data = res.getBody();  
  expect(data.message).toContain('успешно');  
});  
  
test("Проверить сообщения полностью", function() {  
  const data = res.getBody();  
  expect(data.message).toEqual('Авторизация прошла успешно');  
});  
  
test("Проверить тип сообщения строка", function() {  
  const data = res.getBody();  
  expect(data.message).toBe.a('string');  
});
```



```
Params Body * Headers Auth Vars Script * Assert Tests * Docs

1 test("Проверить ответ сервера", function() {
2   expect(res.getStatus()).to.equal(200);
3 });
4
5 test("Проверить сообщения частично", function() {
6   const data = res.getBody();
7   expect(data.message).to.contains('успешно');
8 });
9
10 test("Проверить сообщения полностью", function() {
11   const data = res.getBody();
12   expect(data.message).to.equal('Авторизация прошла успешно')
13   ;
14 });
15 test("Проверить тип сообщения строка", function() {
16   const data = res.getBody();
17   expect(data.message).to.be.a('string');
18 });
```

10. Выполним запрос



POST Test02 x +

POST http://localhost/api/auth.php

Params Body * Headers Auth Vars Script * Assert Tests * Docs

```
1 test("Проверить ответ сервера", function() {
2   expect(res.getStatus()).to.equal(200);
3 });
4
5 test("Проверить сообщения частично", function() {
6   const data = res.getBody();
7   expect(data.message).to.contains('успешно');
8 });
9
10 test("Проверить сообщения полностью", function() {
11   const data = res.getBody();
12   expect(data.message).to.equal('Авторизация прошла успешно');
13 });
14
15 test("Проверить тип сообщения строка", function() {
16   const data = res.getBody();
17   expect(data.message).to.be.a('string');
18 });
```

Response Headers⁸ Timeline Tests⁴

200 OK 3ms 82B

Tests (4/4), Passed: 4, Failed: 0

- ✓ Проверить ответ сервера
- ✓ Проверить сообщения частично
- ✓ Проверить сообщения полностью
- ✓ Проверить тип сообщения строка

Assertions (0/0), Passed: 0, Failed: 0

В результате будет проверено сообщение частично и полностью, а так же проверен тип поля.

11. Запуск тестов из командной строки

Скачайте и установите [NodeJS](#)

Откройте консоль и выполните установку

```
npm install -g @usebruno/cli
```

```
Командная строка
Microsoft Windows [Version 10.0.19045.4894]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Catfish>npm install -g @usebruno/cli
npm warn deprecated har-validator@5.1.5: this library is no longer supported

added 284 packages in 1m

33 packages are looking for funding
  run `npm fund` for details

C:\Users\Catfish>
```

Перейдите в каталог, где находится коллекция запросов и выполните следующую команду:

```
bru run test02.bru
```

результат выполнения будет отражен в консоли

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.4894]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

E:\Git\SomovQA\Practice-Automation-Testing-2024\bruno\tests>dir
Том в устройстве E не имеет метки.
Серийный номер тома: 5A79-A271

Содержимое папки E:\Git\SomovQA\Practice-Automation-Testing-2024\bruno\tests

03.10.2024 13:18 <DIR>      .
03.10.2024 13:18 <DIR>      ..
03.10.2024 10:40          113 bruno.json
03.10.2024 13:25          521 Test01.bru
04.10.2024 09:37          1 045 Test02.bru
                3 файлов          1 679 байт
                2 папок      158 880 210 944 байт свободно

E:\Git\SomovQA\Practice-Automation-Testing-2024\bruno\tests>bru run test02.bru
Running Request

test02 (200 OK) - 117 ms
headers: [object Object]
method: POST
url: http://localhost/api/auth.php
  Проверить ответ сервера
  Проверить сообщения частично
  Проверить сообщения полностью
  Проверить тип сообщения строка

Requests: 1 passed, 1 total
Tests: 4 passed, 4 total
Assertions: 0 passed, 0 total
Ran all requests - 117 ms

Requests: 1 passed, 1 total
Tests: 4 passed, 4 total
Assertions: 0 passed, 0 total

E:\Git\SomovQA\Practice-Automation-Testing-2024\bruno\tests>
```

Другие команды

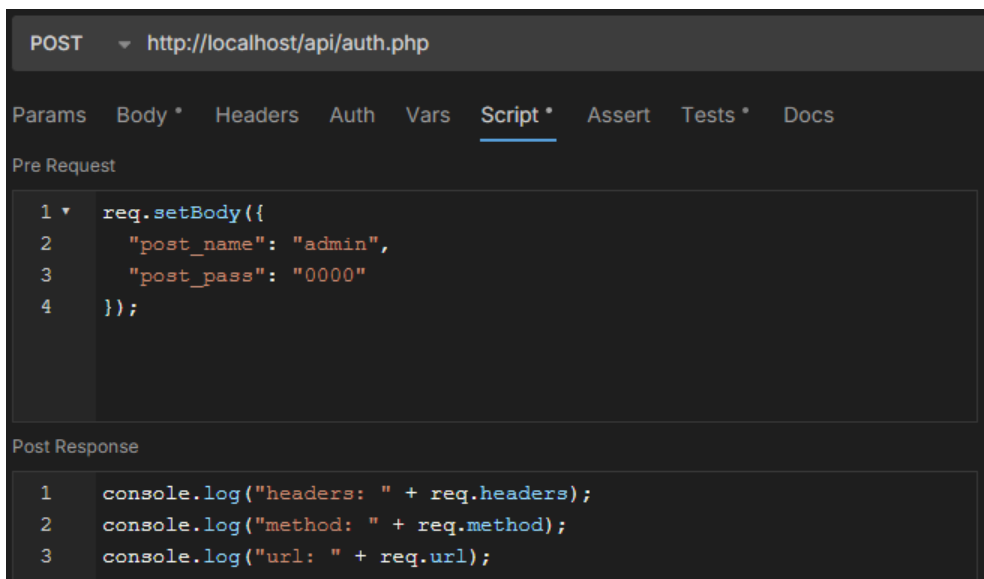
- Чтобы выполнить все запросы в папке, используйте:
`bru run folder`
- Если вам необходимо использовать определенную среду, вы можете передать ее с помощью параметра `--env`:
`bru run folder --env Local`
- Вы можете передавать переменные среды непосредственно в свою коллекцию с помощью параметра `--env-var`:
`bru run folder --env Local --env-var JWT_TOKEN=1234`
- Чтобы сохранить результаты тестов API в файл, используйте опцию `--output`:
`bru run folder --output results.json`
- Для формирования отчета в формате JSON используйте `--reporter-json` опцию:
`bru run request.bru --reporter-json results.json`

- Чтобы создать отчет в формате JUnit, используйте опцию `--reporter-junit`:
`bru run request.bru --reporter-junit results.xml`
- Чтобы создать отчет в формате HTML, понятный человеку, используйте опцию `--reporter-html`:
`bru run request.bru --reporter-html results.html`
- Одновременный запуск нескольких репортеров
`bru run request.bru --reporter-json results.json --reporter-junit results.xml --reporter-html results.html`

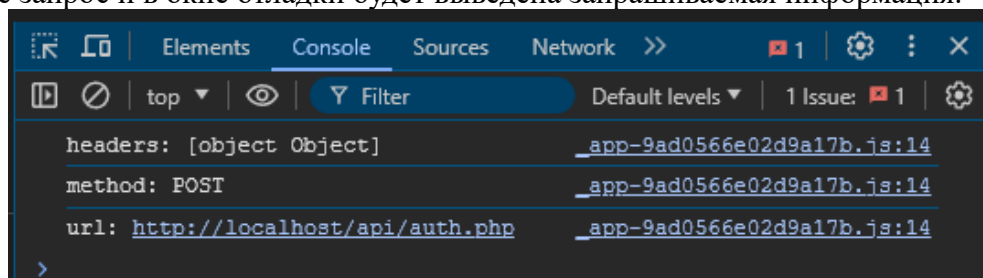
12. Отладка в окне Toggle Developer Tools

На вкладке Script добавьте вывод информации в консоль

```
console.log("headers: " + req.headers);
console.log("method: " + req.method);
console.log("url: " + req.url);
```



Нажмите меню "View" > "Toggle Developer Tools" чтобы открыть окно отладки. Выполните запрос и в окне отладки будет выведена запрашиваемая информация.



Полезные ссылки:

- Официальная страница Bruno
<https://docs.usebruno.com/>
- Официальная документация API Testing
<https://docs.usebruno.com/testing/introduction>
- Официальная документация Scripting
<https://docs.usebruno.com/scripting/getting-started>
- Официальная документация CLI
<https://docs.usebruno.com/bru-cli/overview>

