

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	10
Практика применения JUnit	14
Описание паттернов PageObjects и StepObjects	18
Запуск автотестов с помощью Maven	20
Практика применения TestNG	22
Создание проекта Selenium (Python) с простым автотестом	25
Практика применения Unittest	30
Практика применения PyTest	32
Практика PyTest в стиле xUnit	36
Практика PyTest с применением Fixture	38
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.

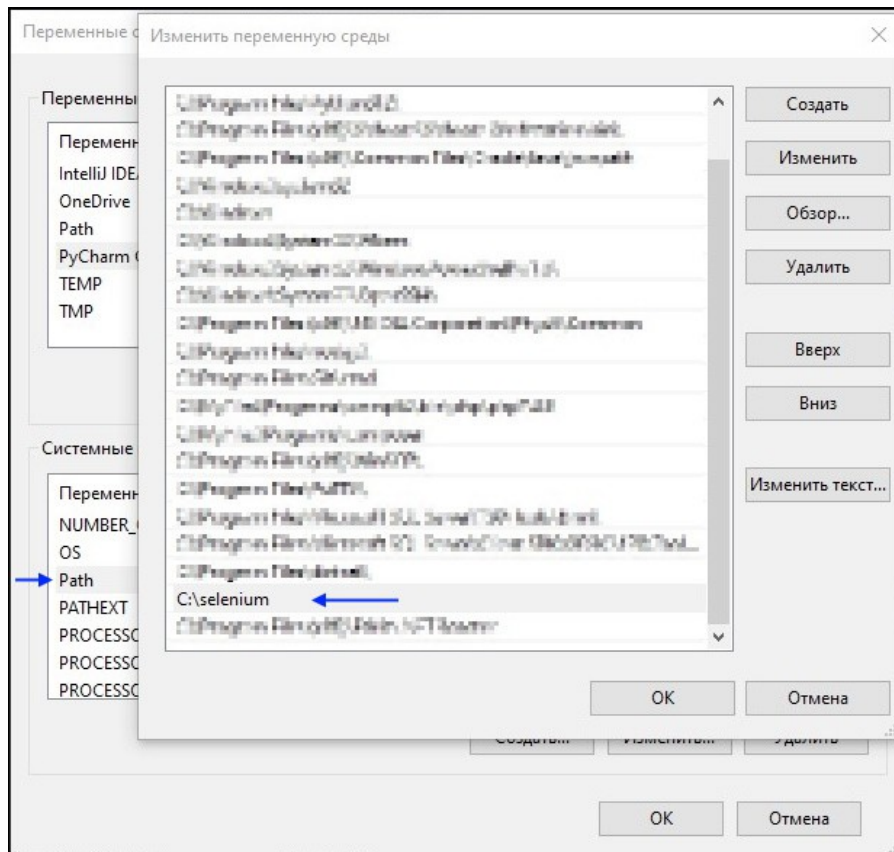
В дополнение практических занятий будет рассмотрено нагрузочное тестирование с помощью JMeter.

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/>

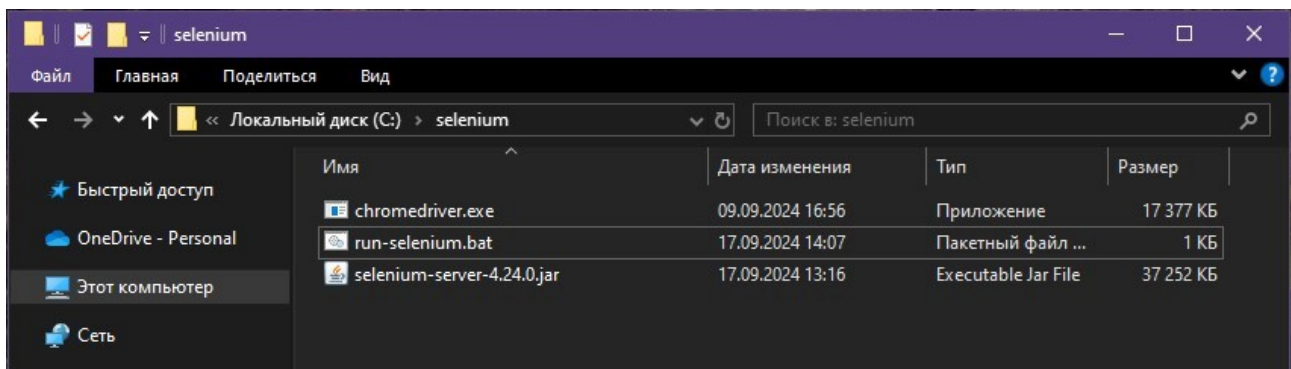
Установка и запуск Selenium Grid

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 mo
de
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 time
s
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 ti
mes
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

Скачать и установить [Visual Studio Code](#)

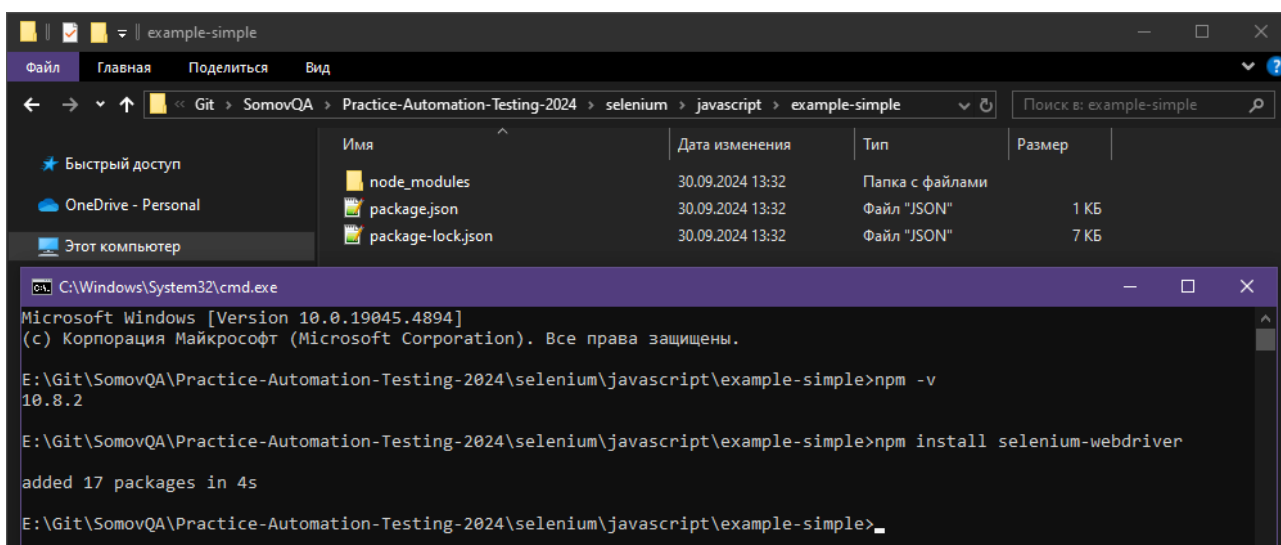
Скачать и установить [NodeJS](#)

Открыть консоль и проверить работу NodeJS с помощью команды:

```
npm -v
```

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

```
npm install selenium-webdriver
```



Открыть папку в редакторе Visual Studio Code

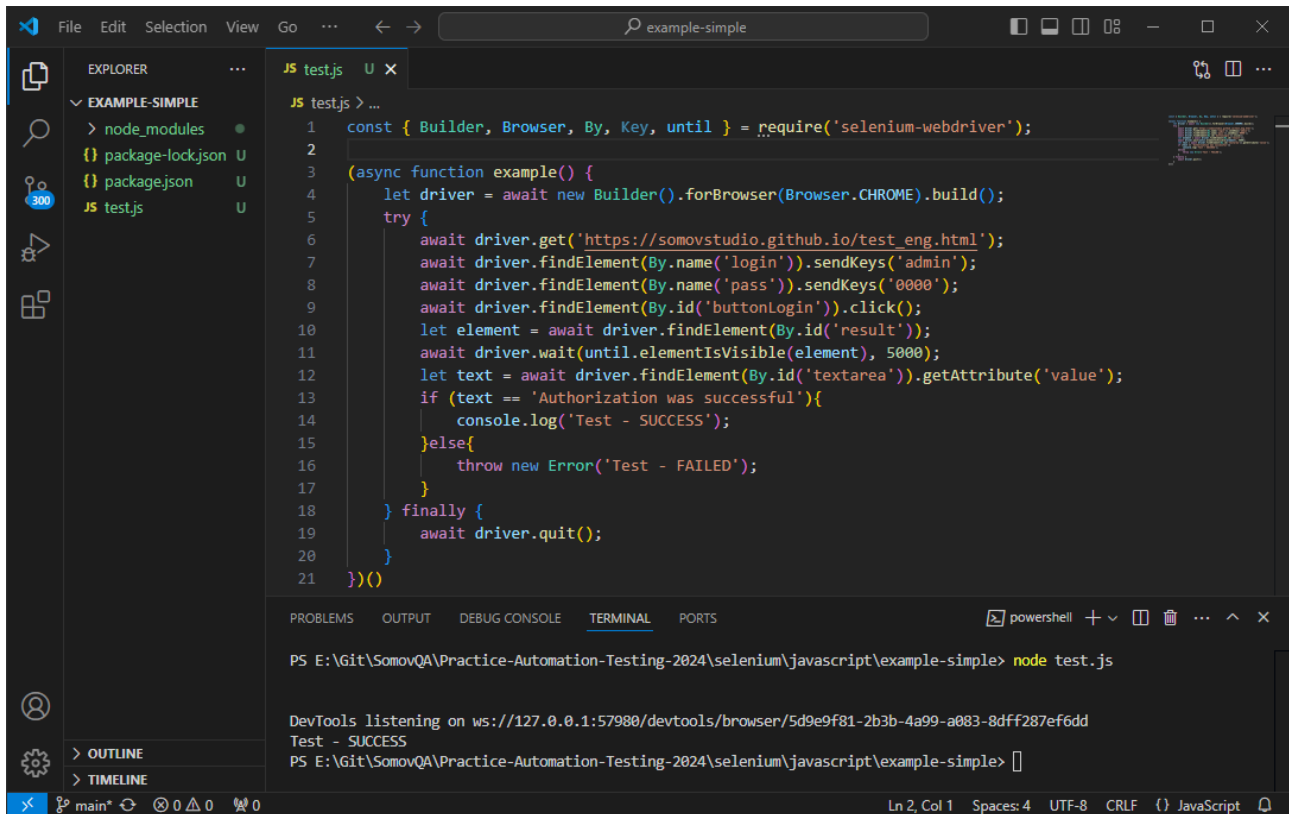
Создать файл test.js и описать автотест следующим образом

```
(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();
  }
})();
```

Запустить автотест командой

```
node test.js
```

Результат автотеста в консоли



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a project named 'EXAMPLE-SIMPLE' with files 'node_modules', '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 perform a login test. The code defines an 'example' async function that navigates to a URL, logs in with 'admin' and '0000', clicks a login button, and checks if the 'result' element contains the text 'Authorization was successful'. The terminal at the bottom shows the command 'node test.js' being executed, with output indicating that DevTools is listening and the test was successful.

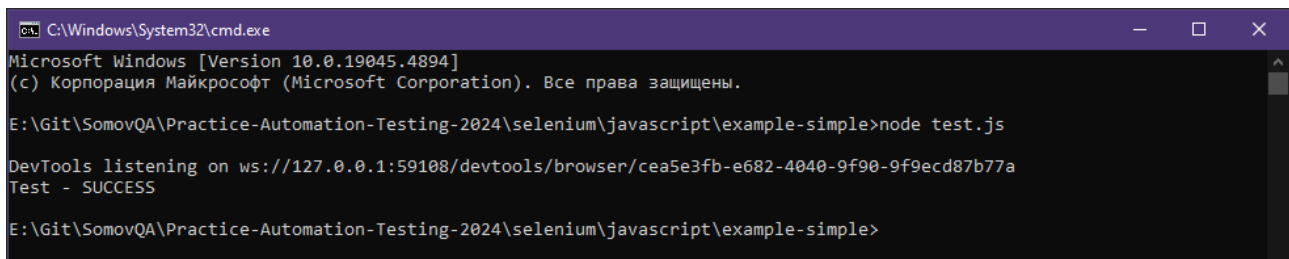
```
1 const { Builder, Browser, By, Key, until } = require('selenium-webdriver');
2
3 (async function example() {
4     let driver = await new Builder().forBrowser(Browser.CHROME).build();
5     try {
6         await driver.get('https://somovstudio.github.io/test_eng.html');
7         await driver.findElement(By.name('login')).sendKeys('admin');
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 titled 'C:\Windows\System32\cmd.exe'. It displays the execution of a Selenium test script. The prompt shows the current directory as 'E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\javascript\example-simple' and the command 'node test.js' being executed. The output shows that DevTools is listening and the test was successful.

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

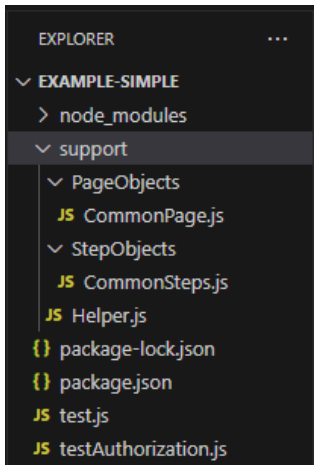
E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\javascript\example-simple>node test.js

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>
```

Описание паттернов PageObjects и StepObjects

Создать папку support со следующей структурой:



Файл Helper.js - описаны константы

```
module.exports = class Helper {  
  static URL = "https://somovstudio.github.io/test_eng.html";  
  static LOGIN = "admin";  
  static PASSWORD = "0000";  
}
```

Файл CommonPage.js - описаны локаторы и статичные методы

```
const { Builder, Browser, By, Key, until } = require('selenium-webdriver');  
  
module.exports = class CommonPage {  
  static nameLogin = "login";  
  static namePassword = "pass";  
  static idButtonLogin = "buttonLogin";  
  static idResult = "result";  
  static idTextarea = "textarea";  
  
  static async getResultText(driver) {  
    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');  
    return text;  
  }  
};
```

Файл CommonSteps.js - описан класс методов для выполнения действий

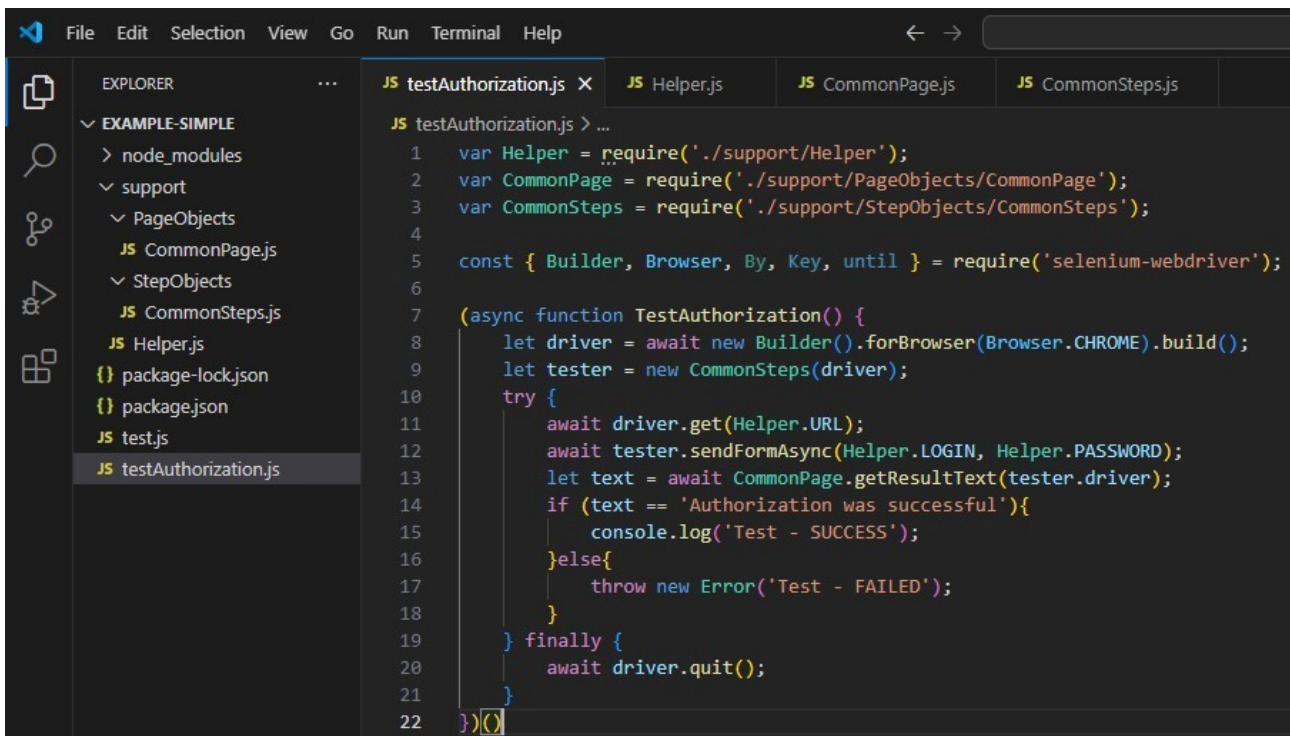
```
const { Builder, Browser, By, Key, until } = require('selenium-webdriver');  
  
module.exports = class CommonSteps {  
  constructor(webdriver) {  
    this.driver = webdriver;  
  }  
  
  async sendFormAsync(login, password) {  
    await this.driver.findElement(By.name('login')).sendKeys('admin');  
    await this.driver.findElement(By.name('pass')).sendKeys('0000');  
    await this.driver.findElement(By.id('buttonLogin')).click();  
  }  
};
```


Файл автотеста testAuthorization.js описать следующим образом:

```
var Helper = require('./support/Helper');
var CommonPage = require('./support/PageObjects/CommonPage');
var CommonSteps = require('./support/StepObjects/CommonSteps');

const { Builder, Browser, By, Key, until } = require('selenium-webdriver');

(async function TestAuthorization() {
  let driver = await new Builder().forBrowser(Browser.CHROME).build();
  let tester = new CommonSteps(driver);
  try {
    await driver.get(Helper.URL);
    await tester.sendFormAsync(Helper.LOGIN, Helper.PASSWORD);
    let text = await CommonPage.getResultText(tester.driver);
    if (text == 'Authorization was successful'){
      console.log('Test - SUCCESS');
    }else{
      throw new Error('Test - FAILED');
    }
  } finally {
    await driver.quit();
  }
})();
```



Запуск автотест командой: `node testAuthorization.js`

Полезные ссылки:

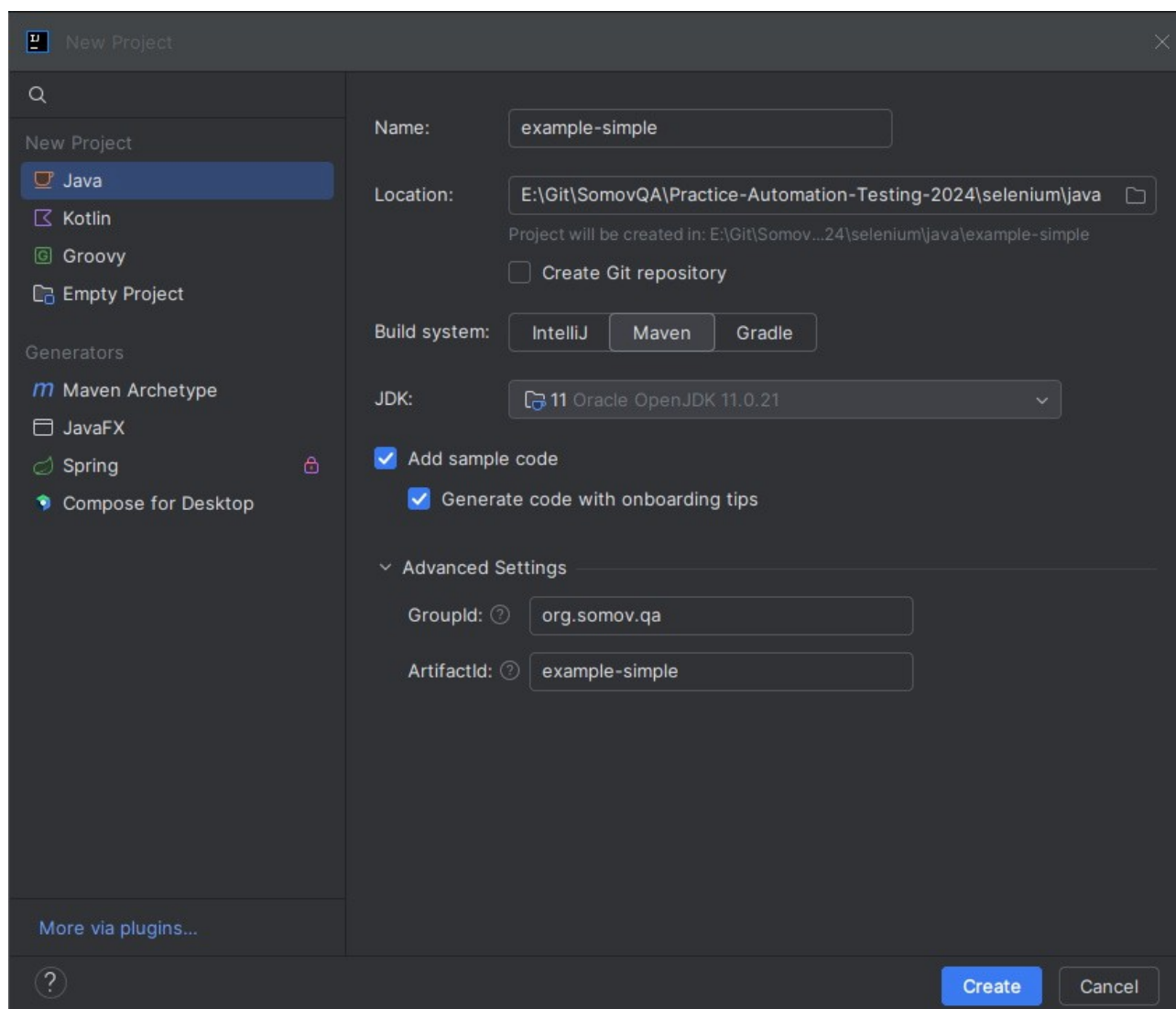
- Официальная документация The Selenium Browser Automation Project
<https://www.selenium.dev/documentation/>
- Документация API Docs
<https://www.selenium.dev/selenium/docs/api/javascript/index.html>
- Официальная страница Visual Studio Code
<https://code.visualstudio.com/>
- Официальная страница NodeJS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

Создание проекта Selenium (Java) с простым автотестом

Скачать и установить [IntelliJ IDEA Community Edition](#)

Скачать и установить [Java SE Development Kit 11](#)

В редакторе IntelliJ IDEA создаем новый проект example-simple выбрав Maven и SDK: Oracle OpenJDK 11



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>
```

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>
```

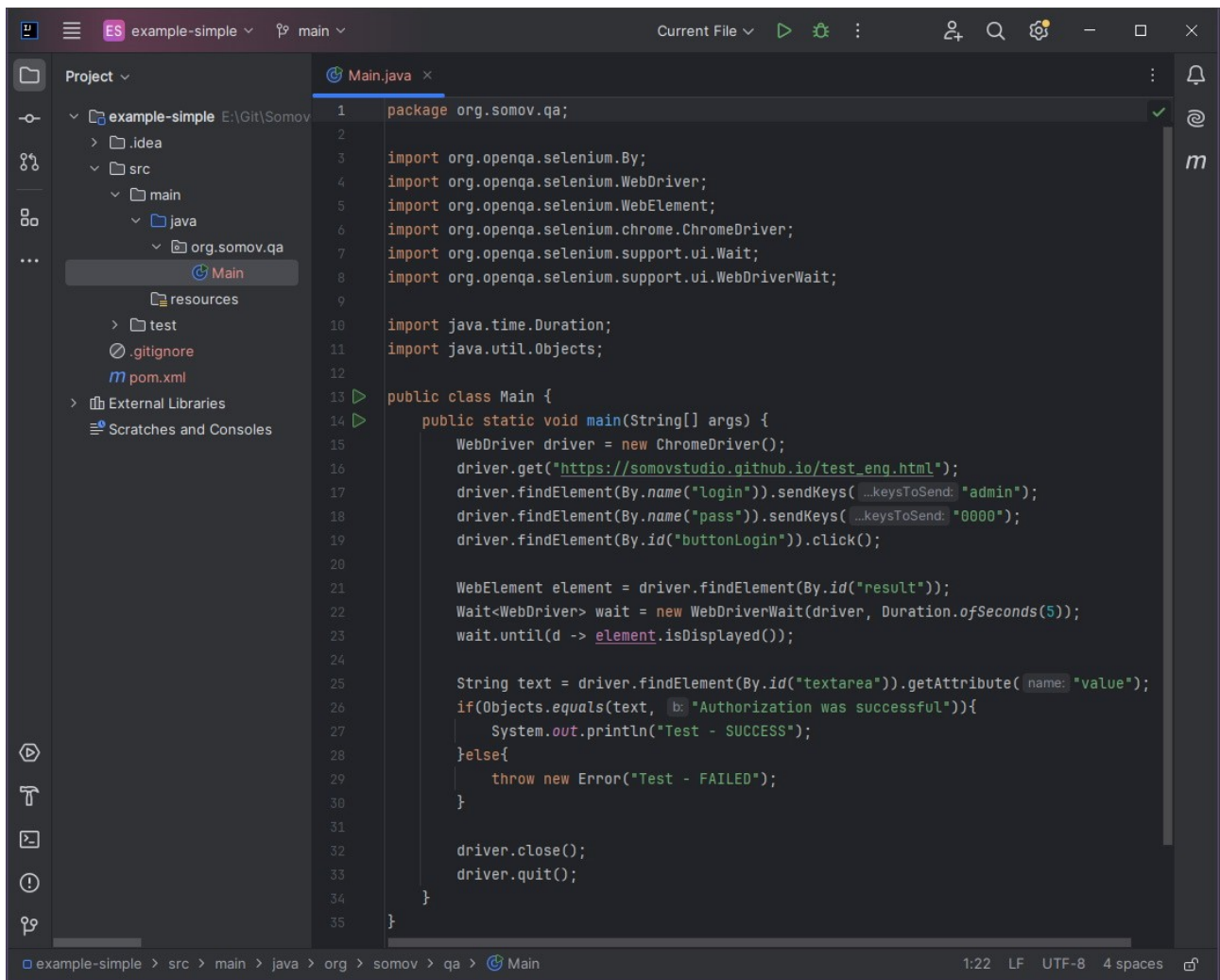
В файле Main.java описать автотест следующим образом

```
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();

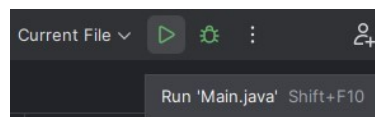
WebElement element = driver.findElement(By.id("result"));
Wait<WebDriver> wait = new WebDriverWait(driver, Duration.ofSeconds(5));
wait.until(d -> element.isDisplayed());

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");
}

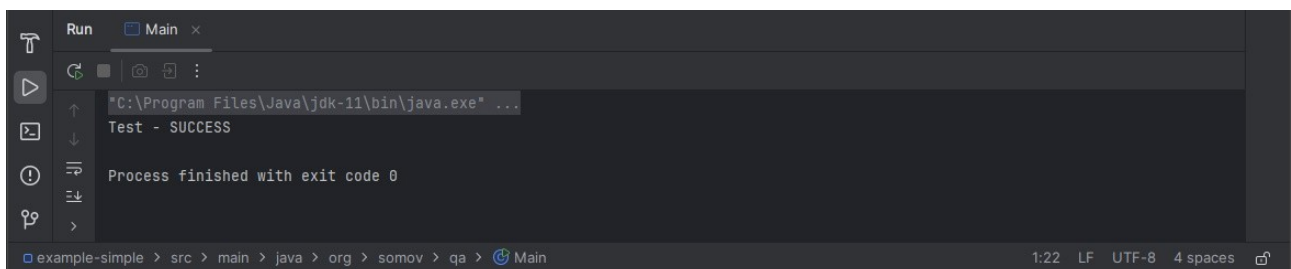
driver.close();
driver.quit();
```



Запустить автотест нажав кнопку Run в IntelliJ IDEA



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

В редакторе IntelliJ IDEA создаем новый проект example-junit

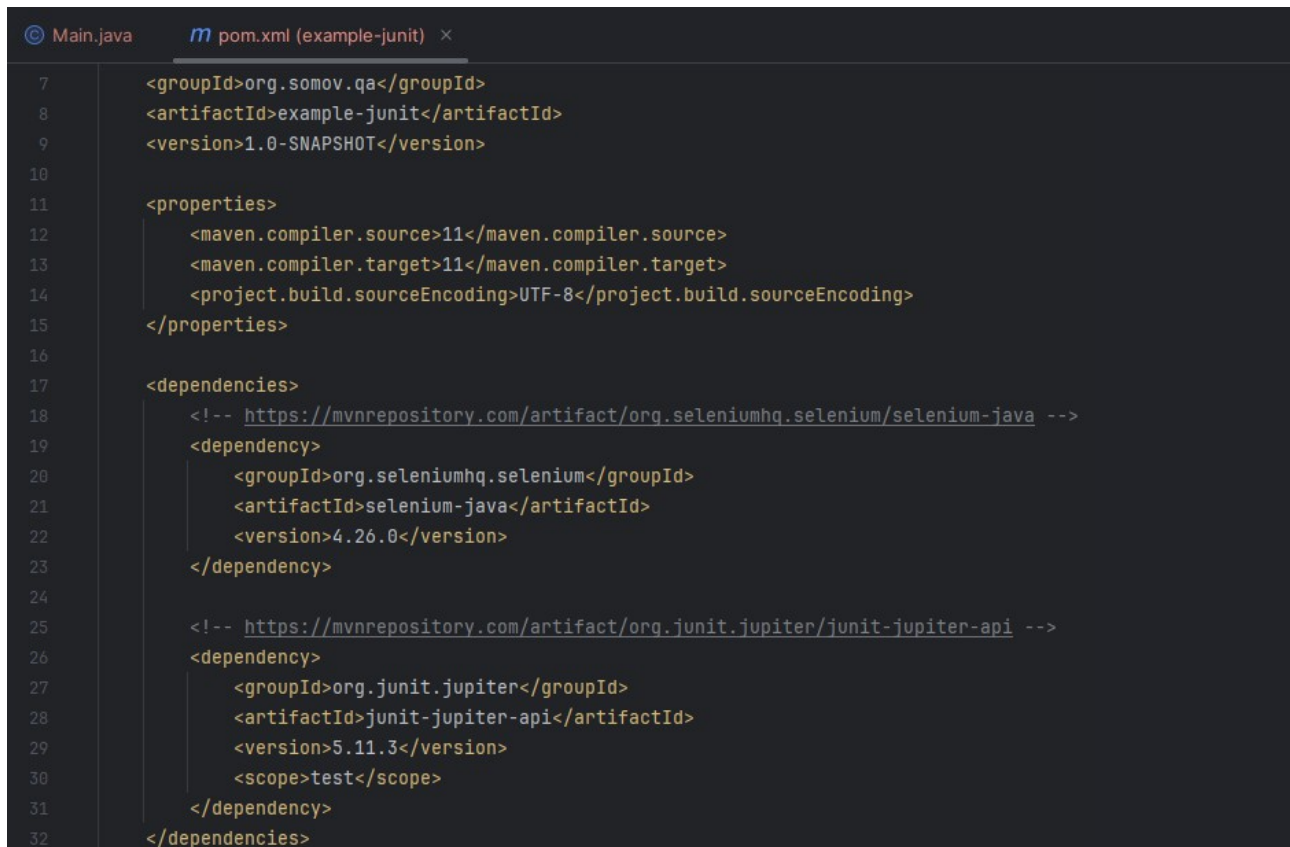
Подключить библиотеку Selenium и JUnit Jupiter API к проекту в файле pom.xml

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

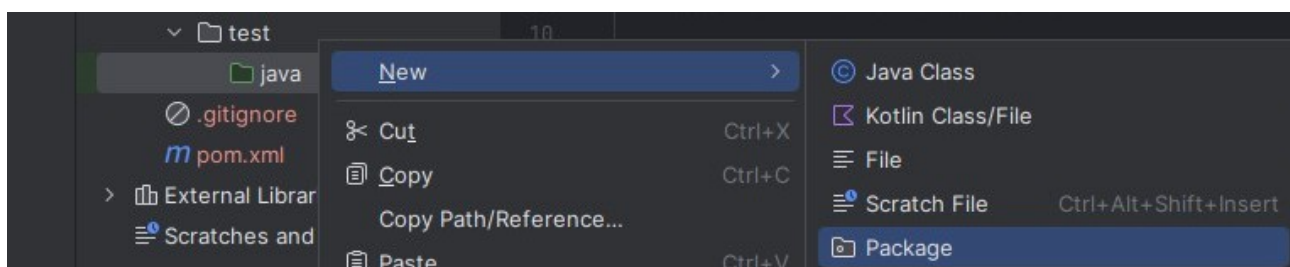
ссылка: <https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api>

```
<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>

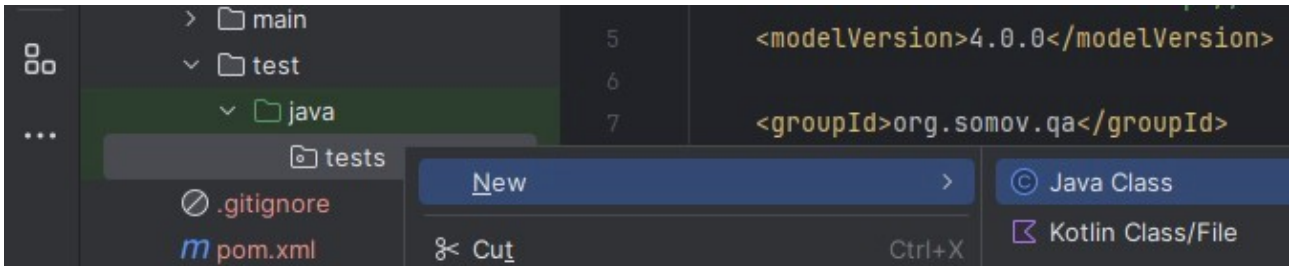
  <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.11.3</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```



В папке test создать пакет tests



В пакете tests создать класс автотеста TestAuthorization.java



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.

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();
    }

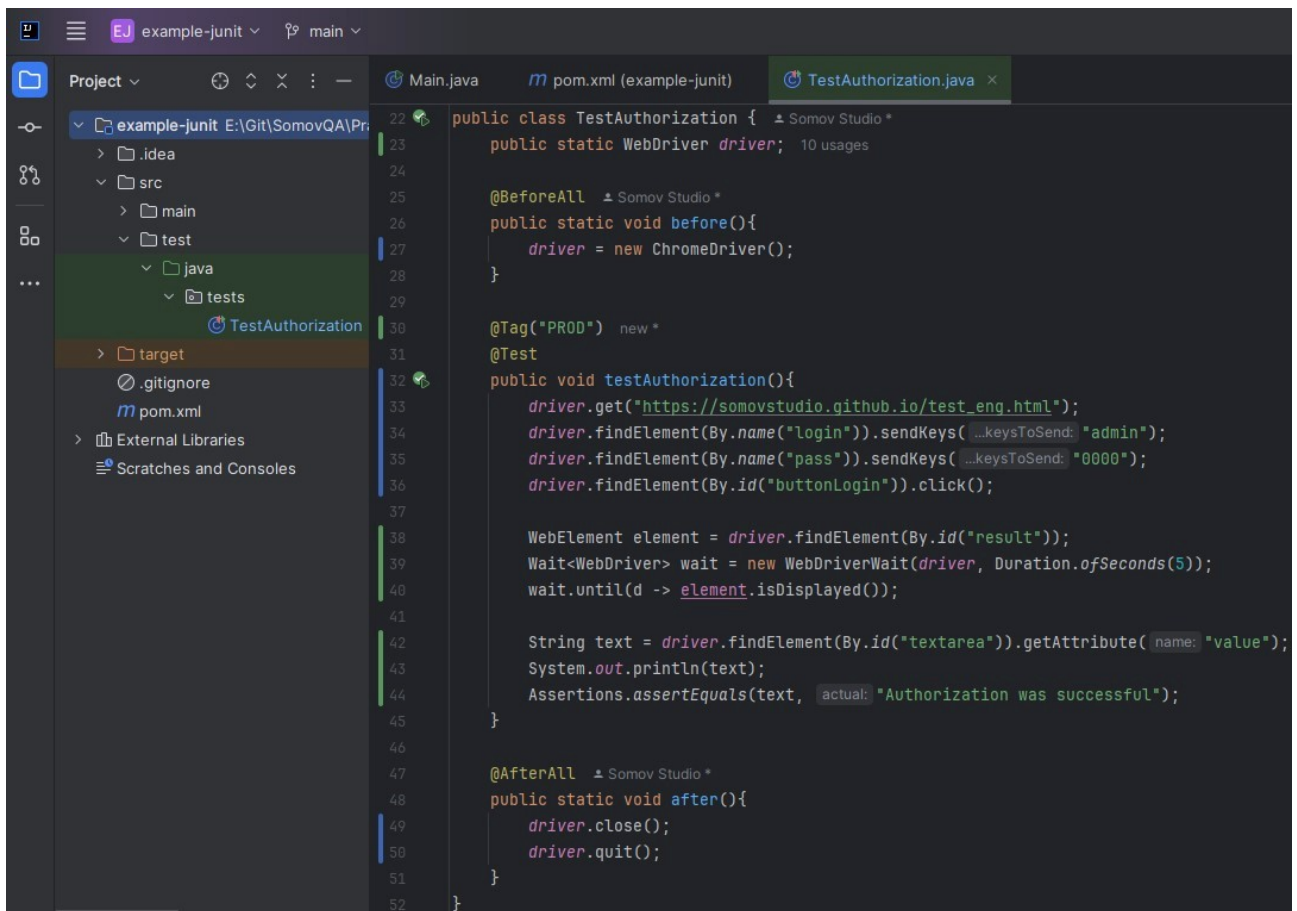
    @Tag("PROD")
    @Test
    public void testAuthorization(){
        driver.get("https://somovstudio.github.io/test_eng.html");
        driver.findElement(By.name("login")).sendKeys("admin");
        driver.findElement(By.name("pass")).sendKeys("0000");
        driver.findElement(By.id("buttonLogin")).click();

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

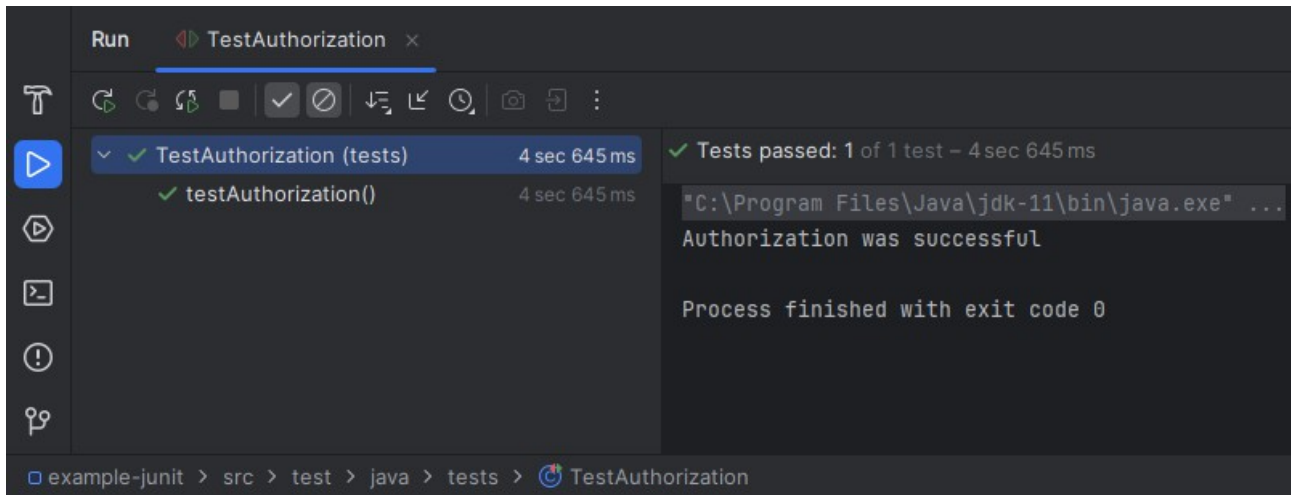
        String text = driver.findElement(By.id("textarea")).getAttribute("value");
        System.out.println(text);
        Assertions.assertEquals(text, "Authorization was successful");
    }

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

```



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

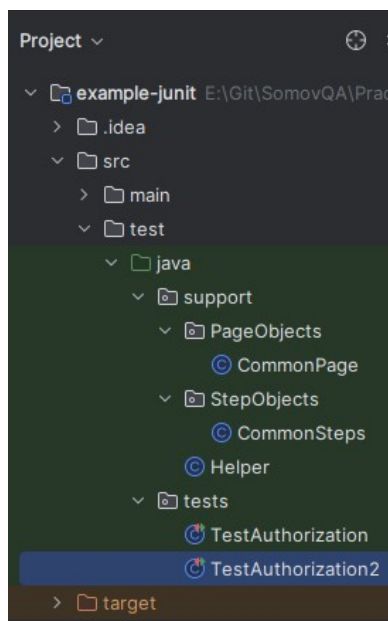


Полезные ссылки:

- 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>

Описание паттернов PageObjects и StepObjects

Описать паттерны PageObjects и StepObjects организовав структуру пакета support следующим образом:



Файл: CommonPage.java

```
package support.PageObjects;

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;

public class CommonPage {
    public static String nameLogin = "login";
    public static String namePassword = "pass";
    public static String idButtonLogin = "buttonLogin";
    public static String idResult = "result";
    public static String idTextarea = "textarea";

    public static String getResultText(WebDriver driver) {
        WebElement element = driver.findElement(By.id(CommonPage.
            idResult));
        Wait<WebDriver> wait = new WebDriverWait(driver,
            Duration.ofSeconds(5));
        wait.until(d -> element.isDisplayed());
        String text = driver.findElement(By.id(CommonPage.
            idTextarea)).getAttribute("value");
        System.out.println("Get message: " + text);
        return text;
    }
}
```

Файл: CommonSteps.java

```
package support.StepObjects;

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 support.PageObjects.CommonPage;

public class CommonSteps {
    public final WebDriver driver;

    public CommonSteps(WebDriver webdriver) {
        driver = webdriver;
    }

    public void sendForm(String login, String password) {
        driver.findElement(
            By.name(CommonPage.nameLogin)).sendKeys(login);
        driver.findElement(
            By.name(CommonPage.namePassword)).sendKeys(password);
        driver.findElement(
            By.id(CommonPage.idButtonLogin)).click();
    }
}
```

В файле Helper.java описаны глобальные константы

```
package support;

public class Helper {
    public static String URL = "https://somovstudio.github.io/test_eng.html";
    public static String LOGIN = "admin";
    public static String PASSWORD = "0000";
}
```

В файле TestAuthorization2.java описать автотест используя паттерны

```
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.chrome.ChromeDriver;

import support.PageObjects.CommonPage;
import support.StepObjects.CommonSteps;
import support.Helper;

public class TestAuthorization2 {

    public static CommonSteps tester;

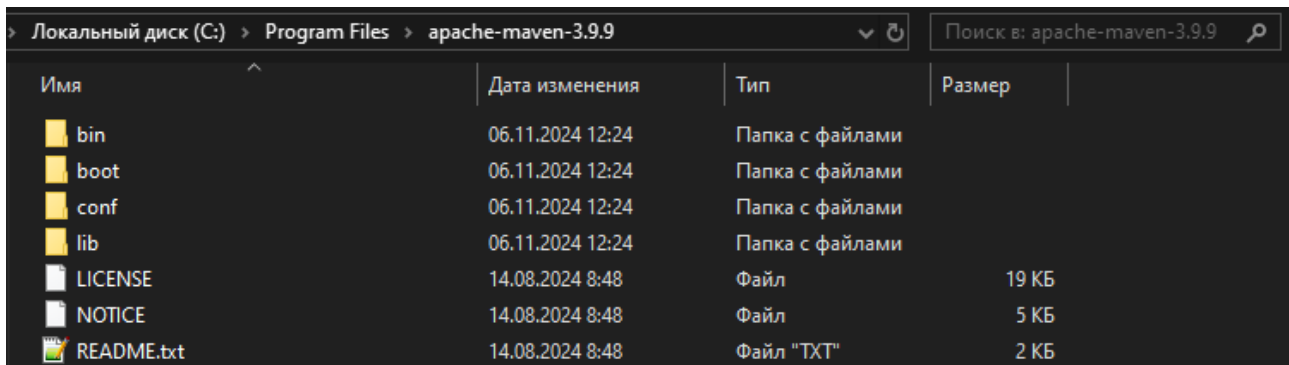
    @BeforeAll
    public static void before() {
        tester = new CommonSteps(new ChromeDriver());
        tester.driver.manage().window().maximize();
    }

    @Tag("PROD")
    @Test
    public void testAuthorization() {
        tester.driver.get(Helper.URL);
        tester.sendForm(Helper.LOGIN, Helper.PASSWORD);
        String text = CommonPage.getResultText(tester.driver);
        Assertions.assertEquals(text, "Authorization was successful");
    }

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

Запуск автотестов с помощью Maven

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

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>

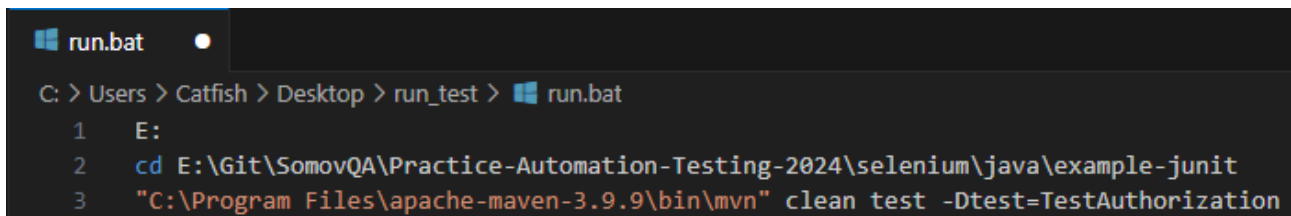
```
<properties>
  <maven.compiler.source>11</maven.compiler.source>
  <maven.compiler.target>11</maven.compiler.target>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
  <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->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

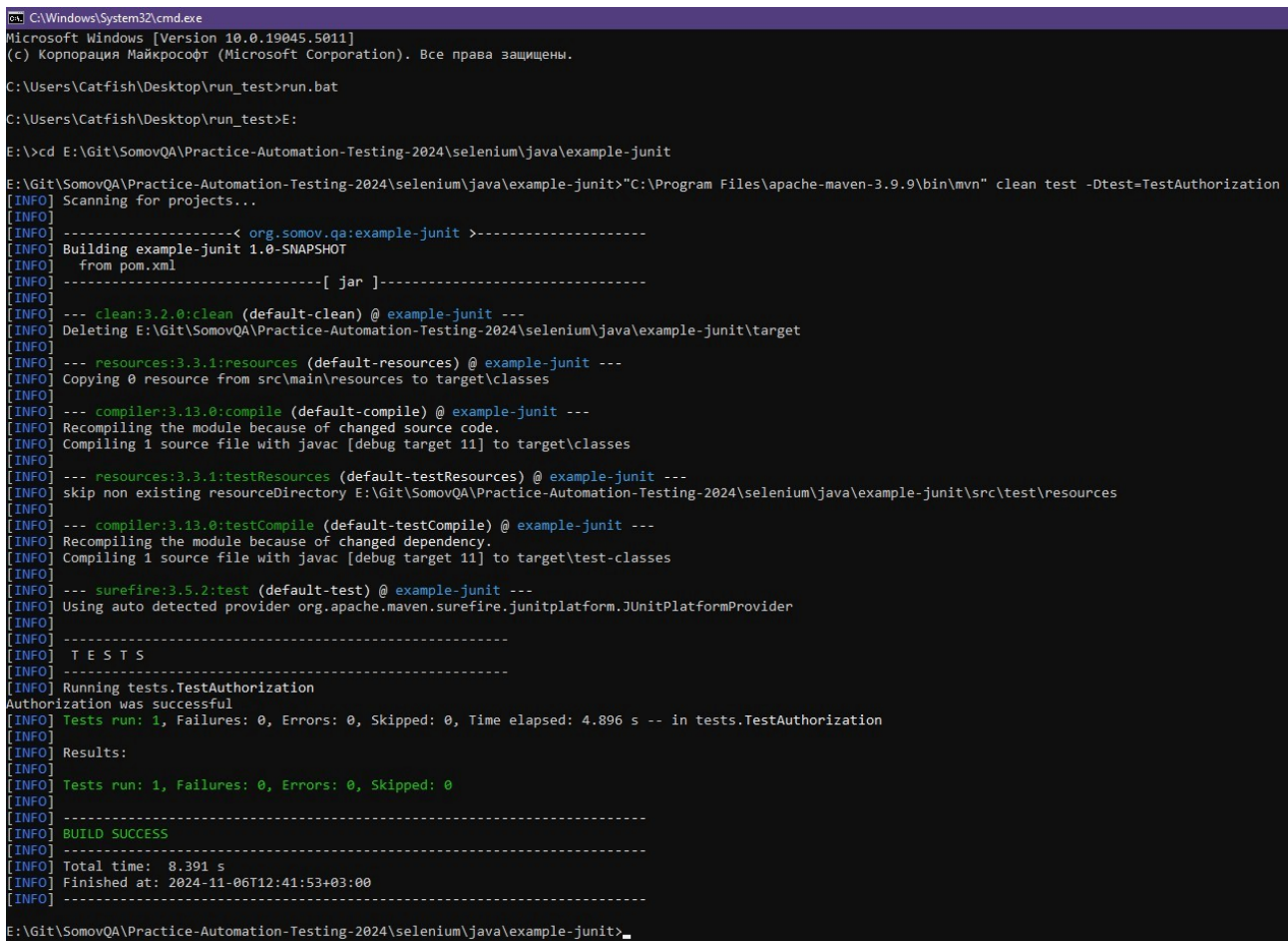
  <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-engine -->
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>5.11.3</version>
    <scope>test</scope>
  </dependency>
</dependencies>
<build>
  <plugins>
    <plugin>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

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
```



Запустить файл run.bat



Практика применения TestNG

В редакторе IntelliJ IDEA создаем новый проект example-testng

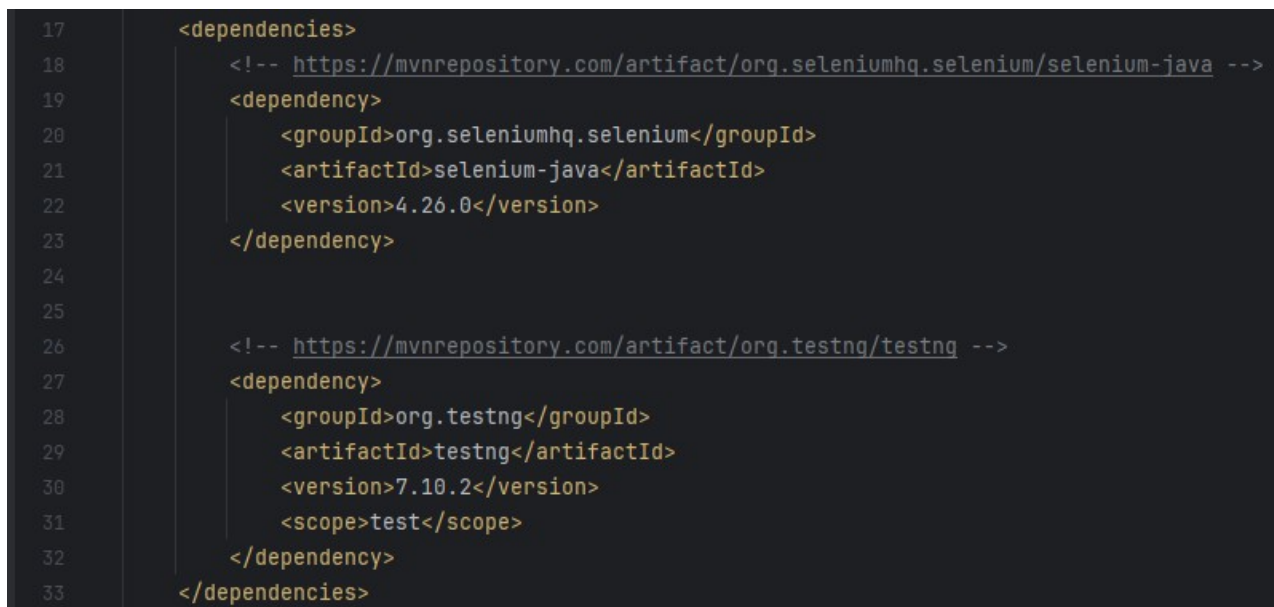
Подключить библиотеку Selenium и TestNG к проекту в файле pom.xml

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

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

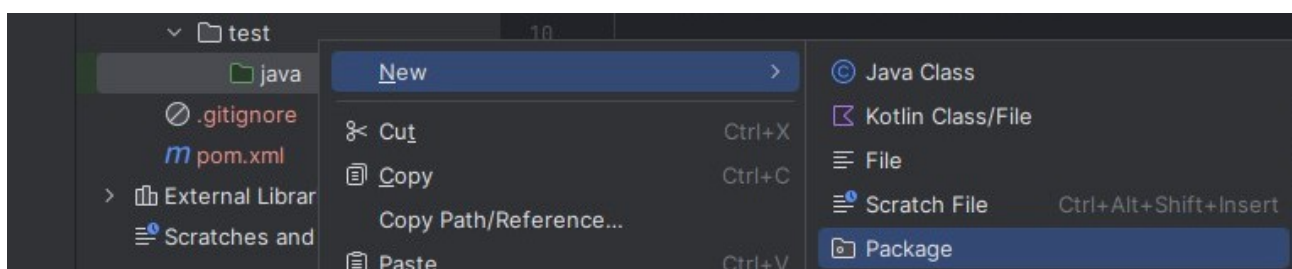
```
<dependencies>
  <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.26.0</version>
  </dependency>

  <!-- 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>
```

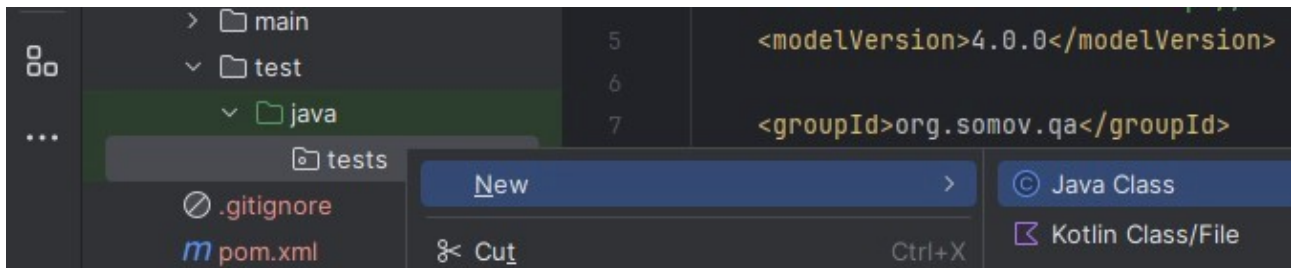


```
17  <dependencies>
18    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
19    <dependency>
20      <groupId>org.seleniumhq.selenium</groupId>
21      <artifactId>selenium-java</artifactId>
22      <version>4.26.0</version>
23    </dependency>
24
25
26    <!-- https://mvnrepository.com/artifact/org.testng/testng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

В папке test создать пакет tests



В пакете tests создать класс автотеста TestAuthorization.java



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;

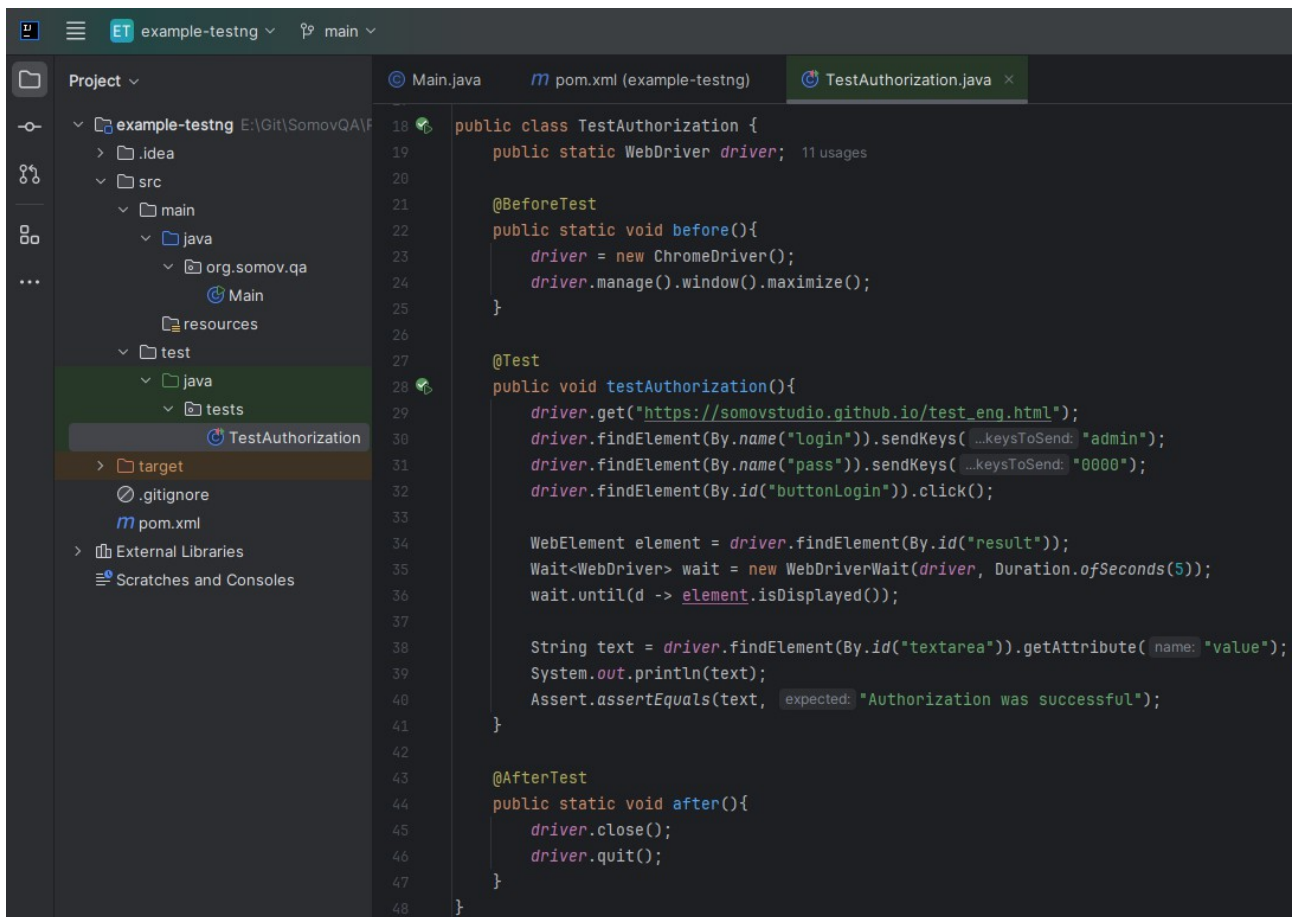
    @BeforeTest
    public static void before(){
        driver = new ChromeDriver();
        driver.manage().window().maximize();
    }

    @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();

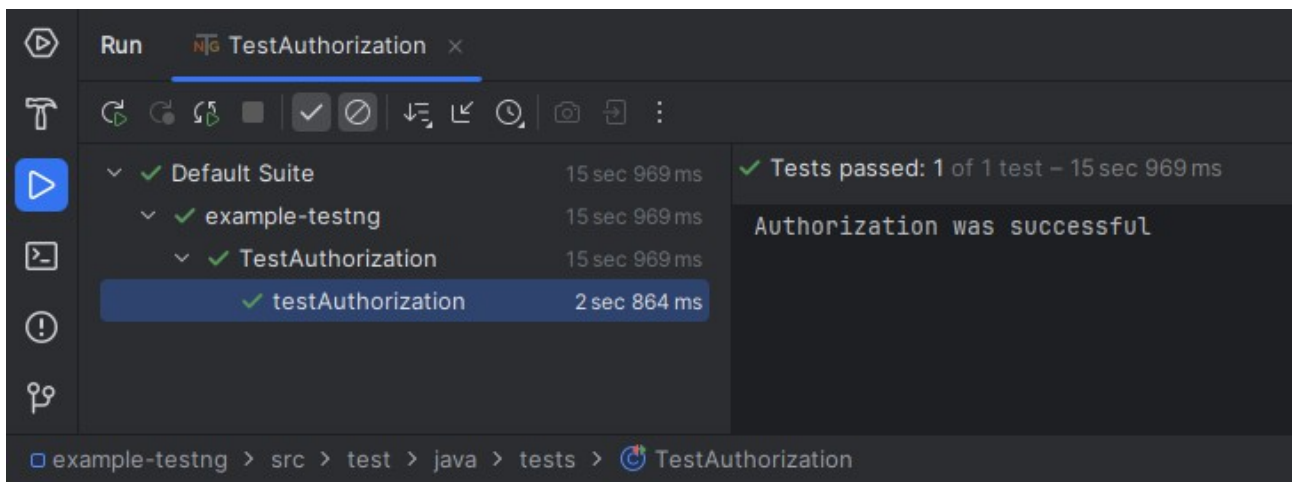
        WebElement element = driver.findElement(By.id("result"));
        Wait<WebDriver> wait = new WebDriverWait(driver, Duration.ofSeconds(5));
        wait.until(d -> element.isDisplayed());

        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();
    }
}
```

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



Полезные ссылки:

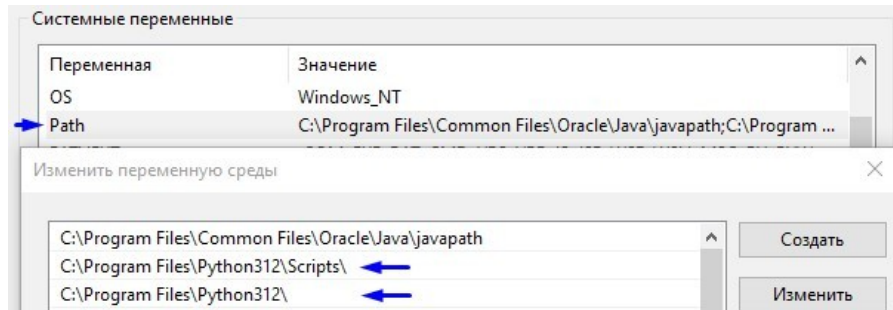
- Selenium Java
<https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>
- TestNG
<https://mvnrepository.com/artifact/org.testng/testng>

Создание проекта Selenium (Python) с простым автотестом

Скачать и установить [Python](#)

В переменной Path прописать путь к папке C:\Program Files\Python

(Панель управления > Система > Дополнительные параметры системы > Переменные среды)



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>
```

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)
```

Проверить установленную версию Selenium

```
pip list
```

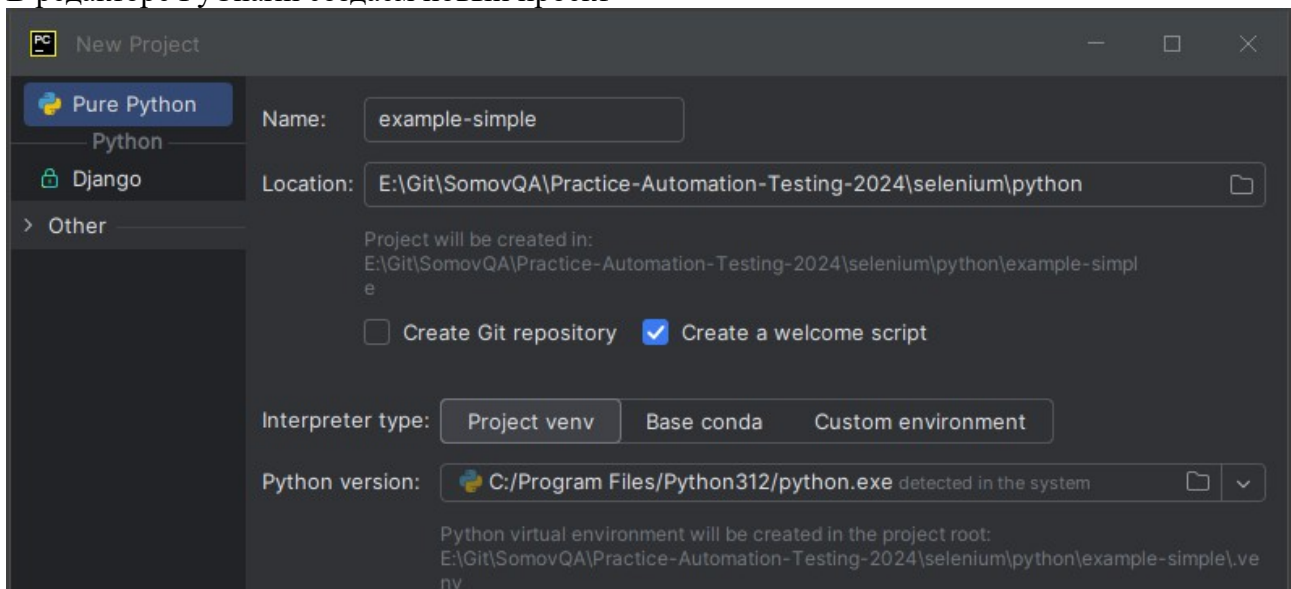
```
Командная строка
Microsoft Windows [Version 10.0.19045.4894]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

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

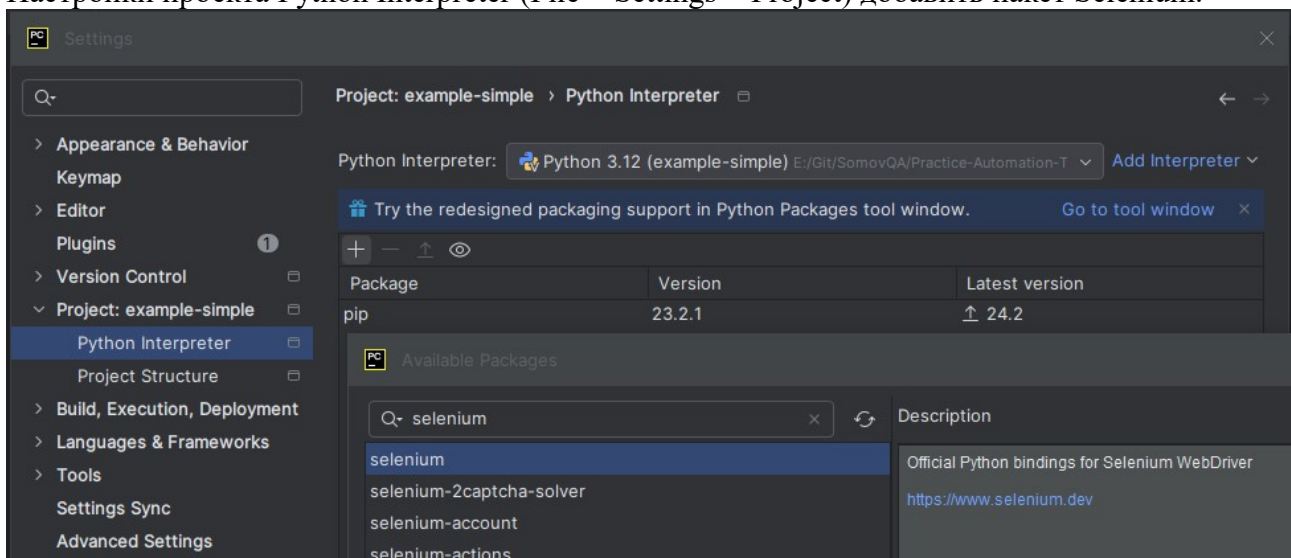
C:\Users\Catfish>
```

Скачать и установить [PyCharm Community Edition](#)

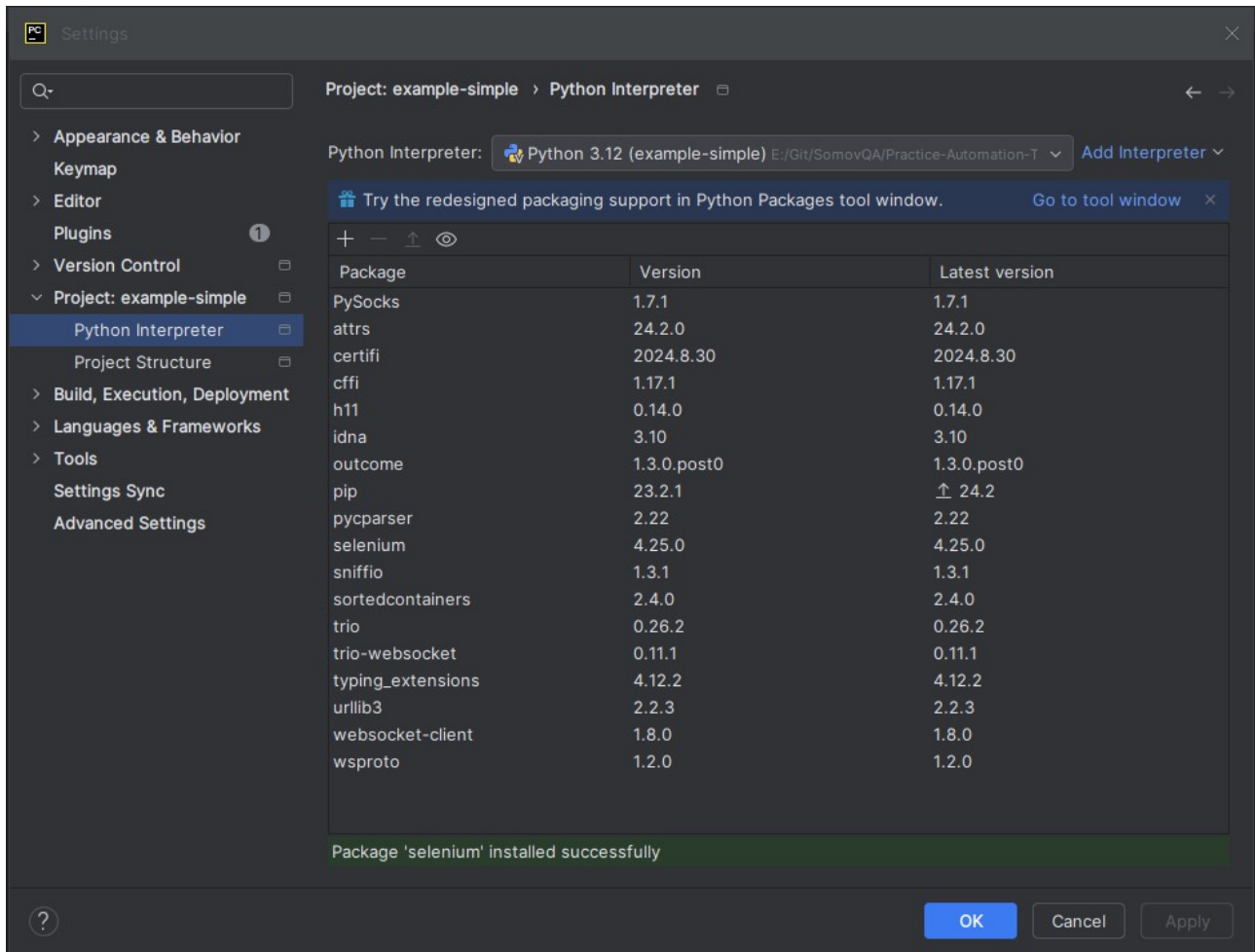
В редакторе PyCharm создаем новый проект



Настройки проекта Python Interpreter (File > Settings > Project) добавить пакет Selenium.



При установке будут добавлены следующие пакеты



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()
```

ПРИМЕРЫ: Ожидание отображения элемента на странице

```
time_wait = 25
locator_popup = "//div[@class='popup_body']"
WebDriverWait(driver, time_wait).until(EC.presence_of_element_located((By.XPATH,
    locator_popup)), message=f"Can't find element by locator {locator_popup}")

WebDriverWait(driver, time_wait).until(EC.presence_of_element_located((By.ID,
    "last_order_sended")), message=f"Can't find element by locator last_order_sended")
```

Функции ожидание отображения элемента

```
def find_element(self, locator, time=10):
    return WebDriverWait(self.driver, time).until(EC.presence_of_element_located(locator),
        message=f"Can't find element by locator {locator}")

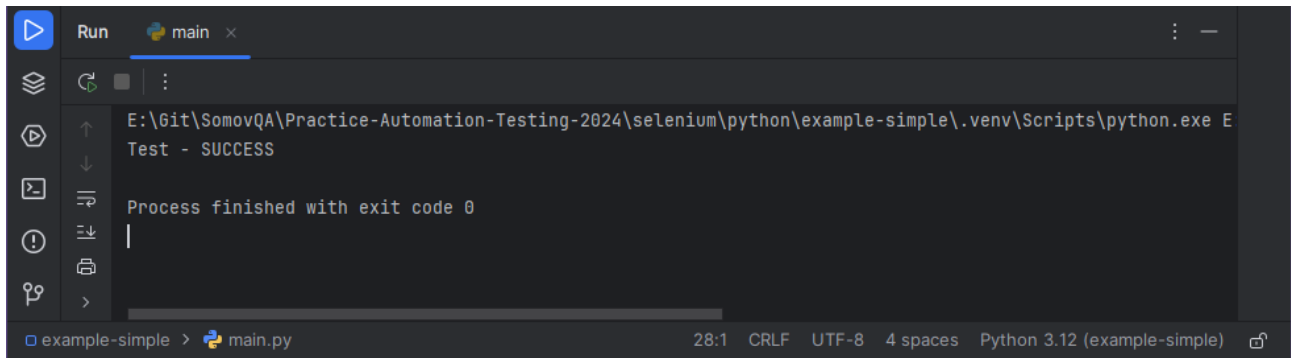
def find_elements(self, locator, time=10):
    return WebDriverWait(self.driver, time).until(EC.presence_of_all_elements_located(locator),
        message=f"Can't find elements by locator {locator}")
```

Простая задержка методом sleep

```
import time

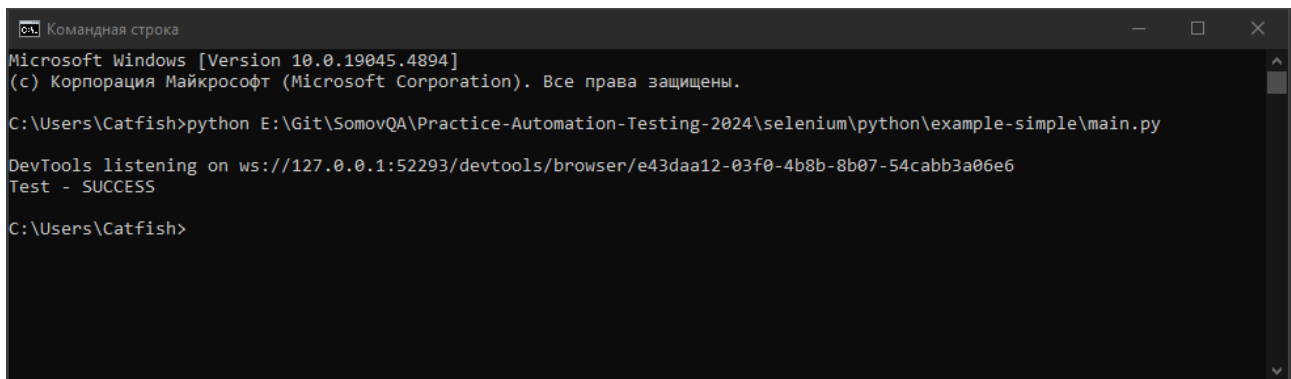
time.sleep(1)
```

Запуск автотеста в редакторе PyCharm



Запуск автотеста из консоли

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

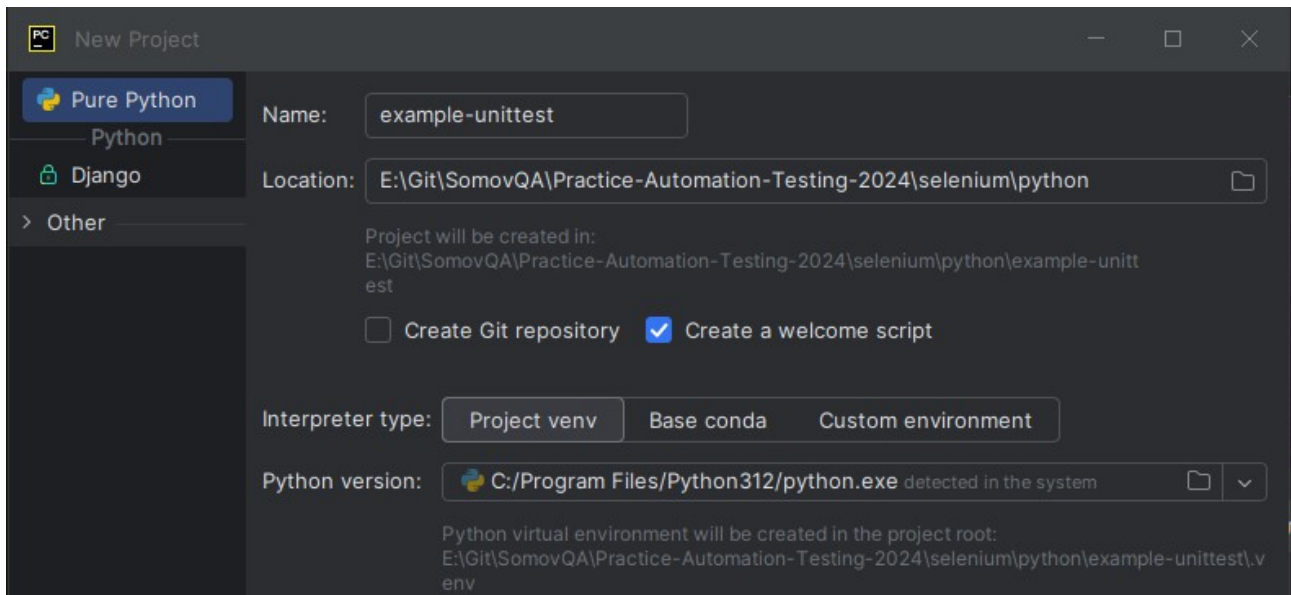


Полезные ссылки:

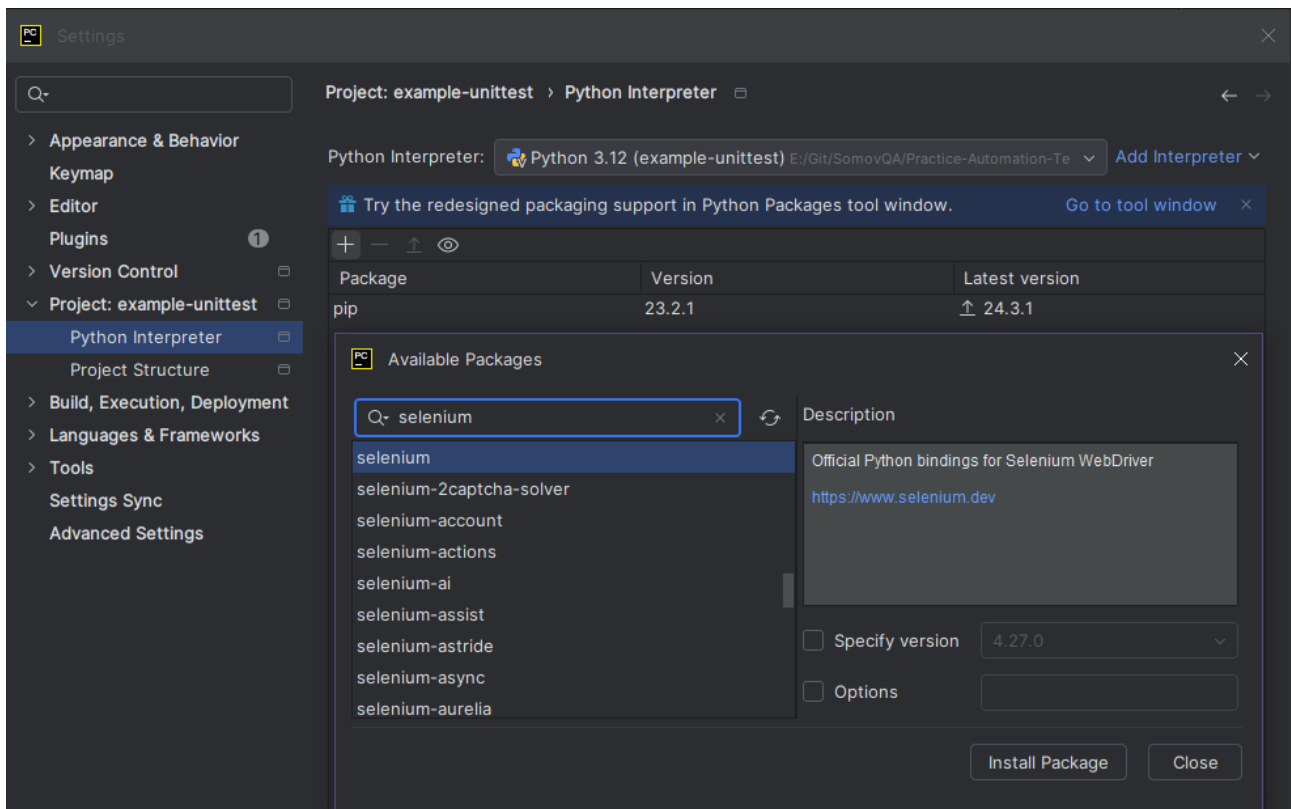
- Официальная документация The Selenium Browser Automation Project
<https://www.selenium.dev/documentation/>
- Документация API Docs (python)
<https://www.selenium.dev/selenium/docs/api/py/index.html>
- Официальная страница PyCharm Community Edition
<https://www.jetbrains.com/pycharm/download/other.html>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

Практика применения Unittest (Python)

В редакторе PyCharm создаем новый проект

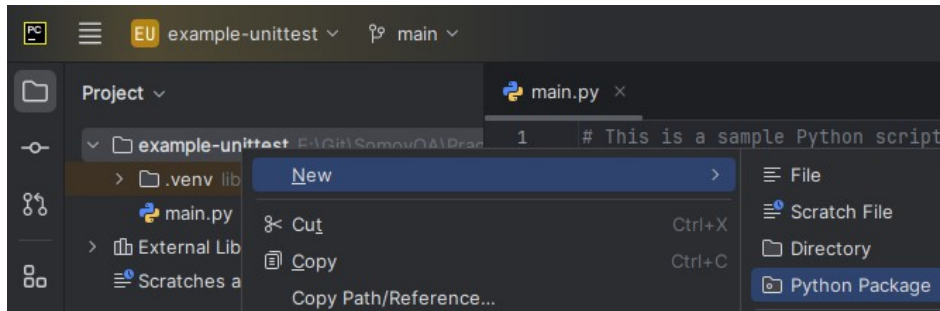


Настройки проекта Python Interpreter (File > Settings > Project) добавить пакет Selenium.

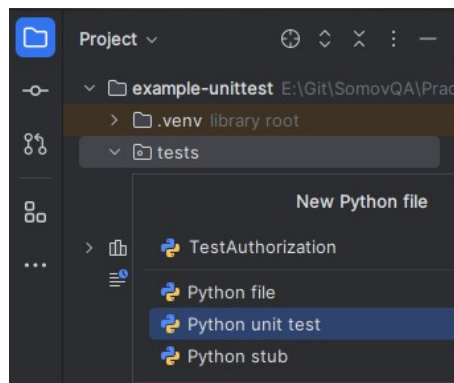


При установке будут добавлены пакеты Selenium

Создать пакет с именем tests



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



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()
```

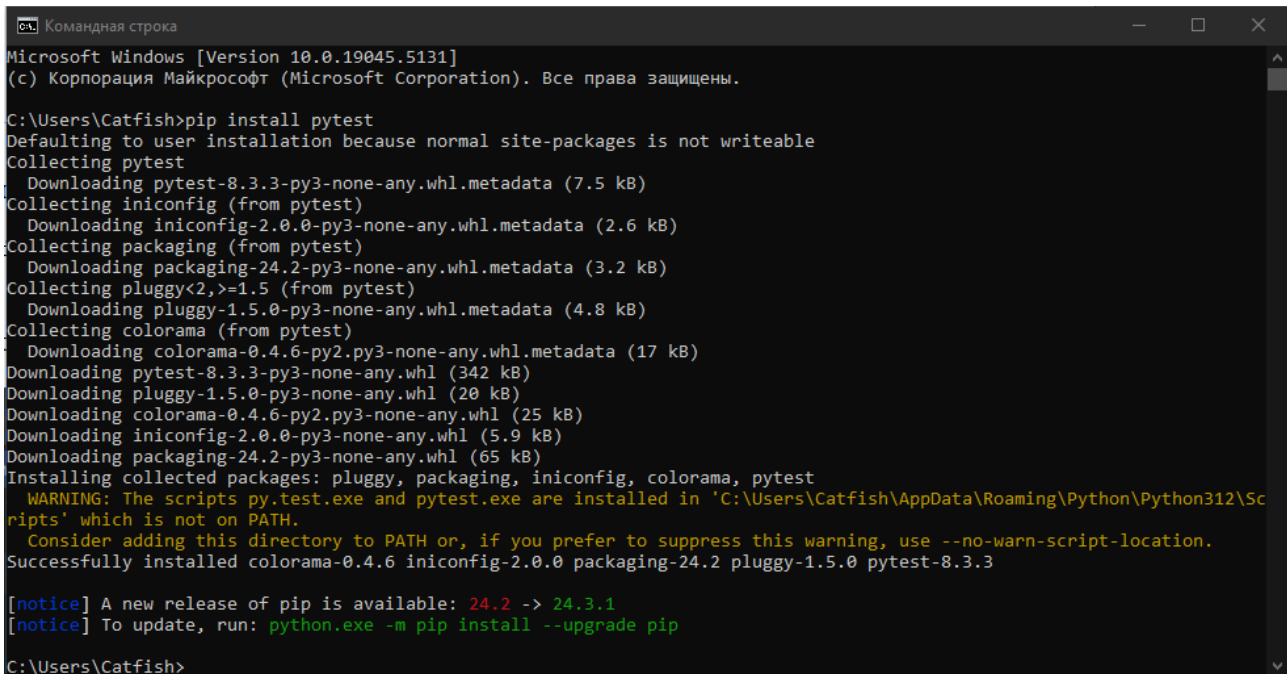
6. Запуск автотеста в редакторе PyCharm или в консоли командой:

```
python E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-unittest\tests\TestAuthorization.py
```

Практика применения PyTest (Python)

Установить PyTest

```
pip install pytest
pip3 install pytest
```



```
Командная строка
Microsoft Windows [Version 10.0.19045.5131]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Catfish>pip install pytest
Defaulting to user installation because normal site-packages is not writeable
Collecting pytest
  Downloading pytest-8.3.3-py3-none-any.whl.metadata (7.5 kB)
Collecting iniconfig (from pytest)
  Downloading iniconfig-2.0.0-py3-none-any.whl.metadata (2.6 kB)
Collecting packaging (from pytest)
  Downloading packaging-24.2-py3-none-any.whl.metadata (3.2 kB)
Collecting pluggy<2,>=1.5 (from pytest)
  Downloading pluggy-1.5.0-py3-none-any.whl.metadata (4.8 kB)
Collecting colorama (from pytest)
  Downloading colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Downloading pytest-8.3.3-py3-none-any.whl (342 kB)
Downloading pluggy-1.5.0-py3-none-any.whl (20 kB)
Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Downloading iniconfig-2.0.0-py3-none-any.whl (5.9 kB)
Downloading packaging-24.2-py3-none-any.whl (65 kB)
Installing collected packages: pluggy, packaging, iniconfig, colorama, pytest
  WARNING: The scripts py.test.exe and pytest.exe are installed in 'C:\Users\Catfish\AppData\Roaming\Python\Python312\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed colorama-0.4.6 iniconfig-2.0.0 packaging-24.2 pluggy-1.5.0 pytest-8.3.3

[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\Catfish>
```

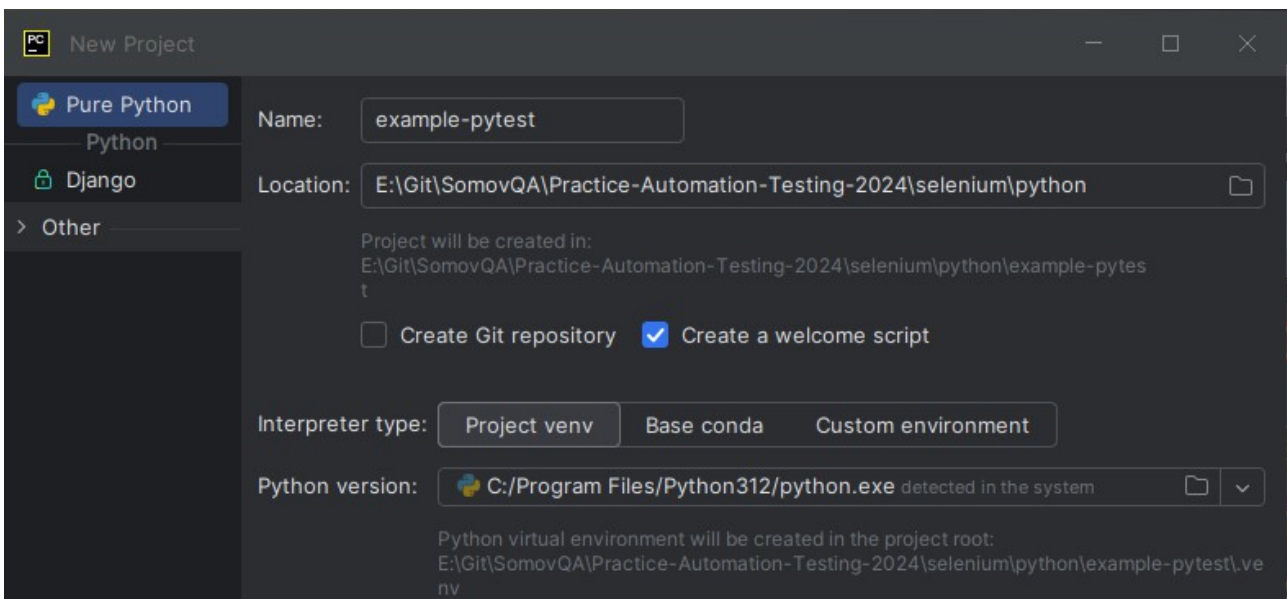
Выполнить обновление если это было предложено

```
python.exe -m pip install --upgrade pip
```

Для проверки выполнить команду

```
pip3 show pytest
pytest --version
```

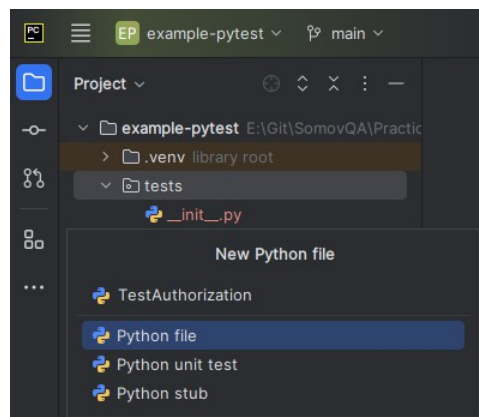
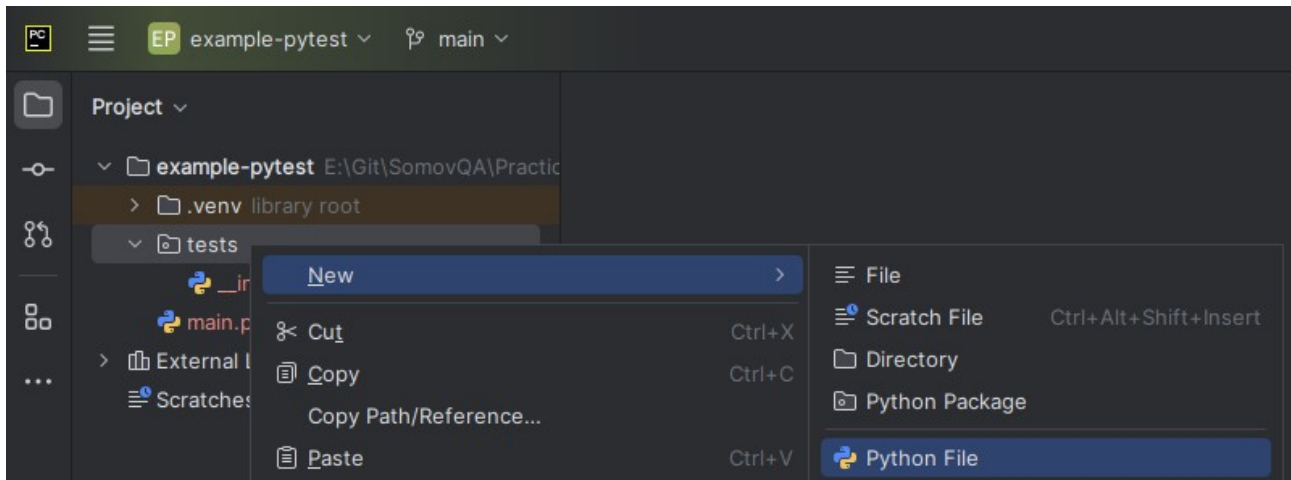
В редакторе PyCharm создаем новый проект



Настройки проекта Python Interpreter (File > Settings > Project) добавить пакет Selenium.
При установке будут добавлены пакеты Selenium

В корне проекта создать пакет с именем tests

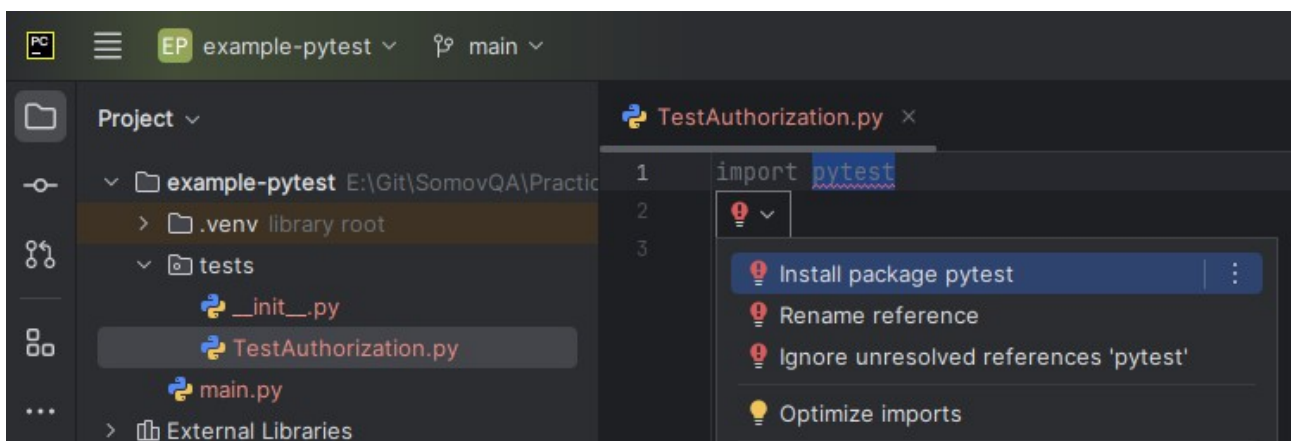
В пакете tests создать файл (Python file) с именем TestAuthorization.py



В файле TestAuthorization.py ввести строку

```
import pytest
```

Выполнить установку пакета pytest как предлагает редактор или через настройки проекта Python Interpreter (File > Settings > Project)



После установки pytest.exe будет находится по адресу \example-pytest\.venv\Scripts

Practice-Automation-Testing-2024 > selenium > python > example-pytest > .venv > Scripts >				
Имя	Дата изменения	Тип	Размер	
.pytest_cache	27.11.2024 14:51	Папка с файлами		
activate	27.11.2024 12:48	Файл	3 КБ	
activate.bat	27.11.2024 12:48	Пакетный файл ...	2 КБ	
activate.fish	27.11.2024 12:48	Файл "FISH"	4 КБ	
activate.nu	27.11.2024 12:48	Файл "NU"	3 КБ	
activate.ps1	27.11.2024 12:48	Сценарий Windo...	2 КБ	
activate_this.py	27.11.2024 12:48	JetBrains PyChar...	2 КБ	
deactivate.bat	27.11.2024 12:48	Пакетный файл ...	1 КБ	
pip.exe	27.11.2024 12:48	Приложение	106 КБ	
pip3.12.exe	27.11.2024 12:48	Приложение	106 КБ	
pip-3.12.exe	27.11.2024 12:48	Приложение	106 КБ	
pip3.exe	27.11.2024 12:48	Приложение	106 КБ	
py.test.exe	27.11.2024 13:20	Приложение	106 КБ	
pydoc.bat	27.11.2024 12:48	Пакетный файл ...	1 КБ	
pytest.exe	27.11.2024 13:20	Приложение	106 КБ	
python.exe	27.11.2024 12:48	Приложение	264 КБ	
pythonw.exe	27.11.2024 12:48	Приложение	253 КБ	
wsdump.exe	27.11.2024 12:55	Приложение	106 КБ	

В файле TestAuthorization.py описать автотест следующим образом

```
import pytest

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

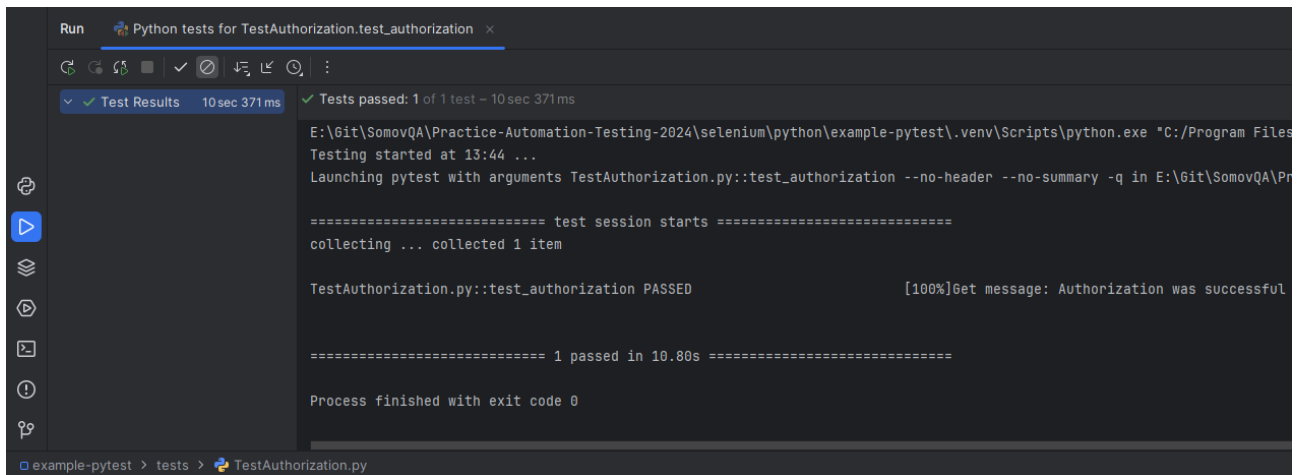
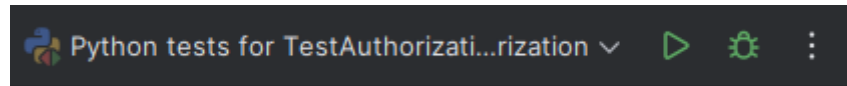
def test_authorization():
    driver = webdriver.Chrome()
    driver.maximize_window()
    driver.get('https://somovstudio.github.io/test_eng.html')
    driver.find_element(By.NAME, 'login').send_keys('admin')
    driver.find_element(By.NAME, 'pass').send_keys('0000')
    driver.find_element(By.ID, 'buttonLogin').click()
    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)

    assert text == 'Authorization was successful', "Получено некорректное сообщение"

    driver.close()
    driver.quit()

if __name__ == '__main__':
    pytest.main(["-s", "TestAuthorization.py"])
```

Запуск автотеста в редакторе PyCharm

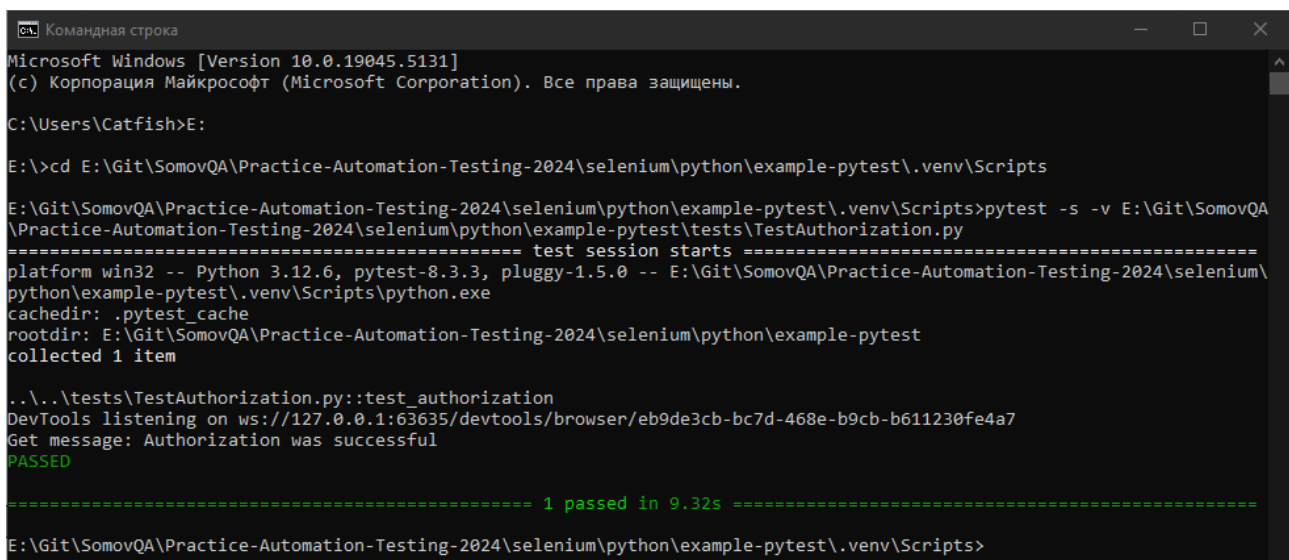


ИЛИ В КОНСОЛИ ВЫПОЛНИТЬ КОМАНДЫ

E:

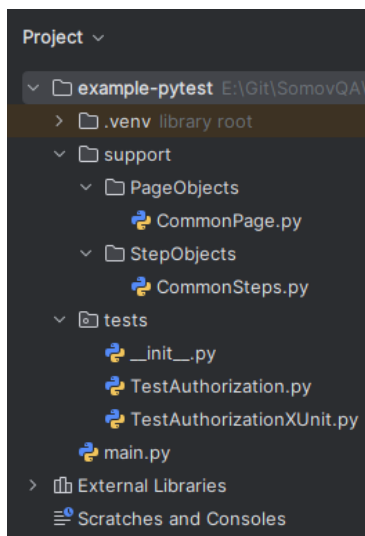
```
cd E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-pytest\.venv\Scripts
```

```
pytest -s -v E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-pytest\tests\TestAuthorization.py
```



Практика PyTest в стиле xUnit

Описание паттернов PageObjects и StepObjects



В файле CommonPage.py

```
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait

class CommonPage:
    nameLogin = "login"
    namePassword = "pass"
    idButtonLogin = "buttonLogin"
    idResult = "result"
    idTextarea = "textarea"

    def getResultText(driver):
        element = driver.find_element(By.ID, CommonPage.idResult)
        wait = WebDriverWait(driver, timeout=5)
        wait.until(lambda d: element.is_displayed())
        text = driver.find_element(By.ID,
                                   CommonPage.idTextarea).get_property('value')
        print("Get message: " + text)
        return text
```

В файле CommonSteps.py

```
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

from support.PageObjects.CommonPage import CommonPage

class CommonSteps:

    def __init__(self, webdriver):
        self.driver = webdriver

    def sendForm(self, login, password):
        self.driver.find_element(By.NAME,
                                 CommonPage.nameLogin).send_keys(login)
        self.driver.find_element(By.NAME,
                                 CommonPage.namePassword).send_keys(password)
        self.driver.find_element(By.ID,
                                 CommonPage.idButtonLogin).click()
```

В файле TestAuthorizationXUnit.py описать автотест следующим образом

```
import pytest

from selenium import webdriver
from support.PageObjects.CommonPage import CommonPage
from support.StepObjects.CommonSteps import CommonSteps

class TestAuthorizationXUnit:

    def setup_method(self):
        self.driver = webdriver.Chrome()
        self.driver.maximize_window()

    def test(self):
        tester = CommonSteps(self.driver)
        tester.driver.get('https://somovstudio.github.io/test_eng.html')
        tester.sendForm('admin', '0000')
        text = CommonPage.getResultText(tester.driver)
        assert text == 'Authorization was successful', "Получено некорректное сообщение"
```

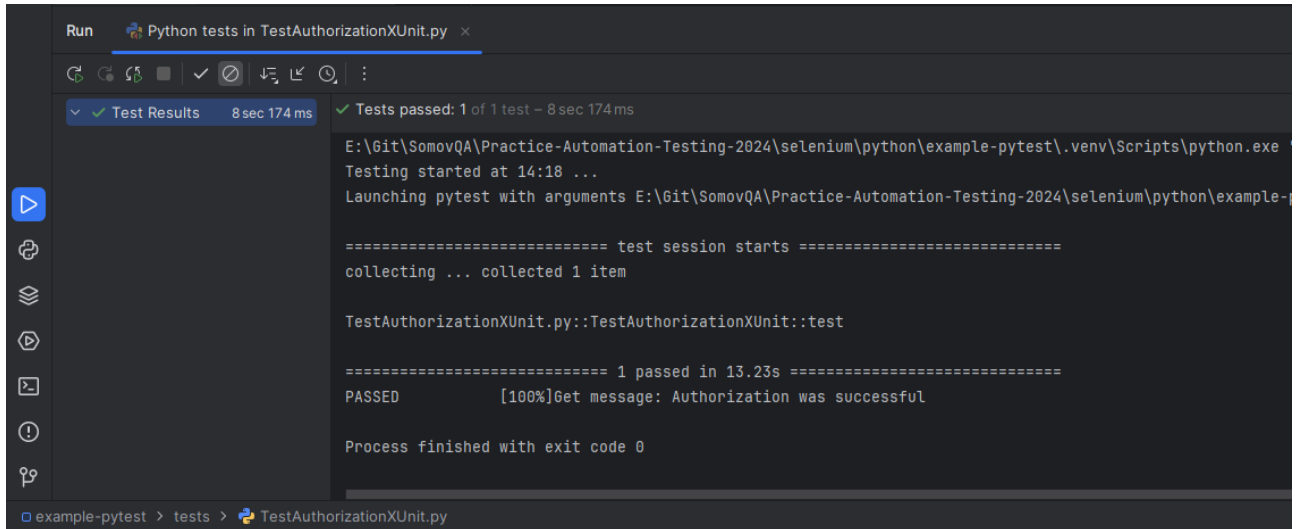
```

def teardown_method(self):
    self.driver.close()
    self.driver.quit()

if __name__ == '__main__':
    pytest.main(["-s", "TestAuthorizationXUnit.py"])

```

Запуск автотеста в редакторе PyCharm

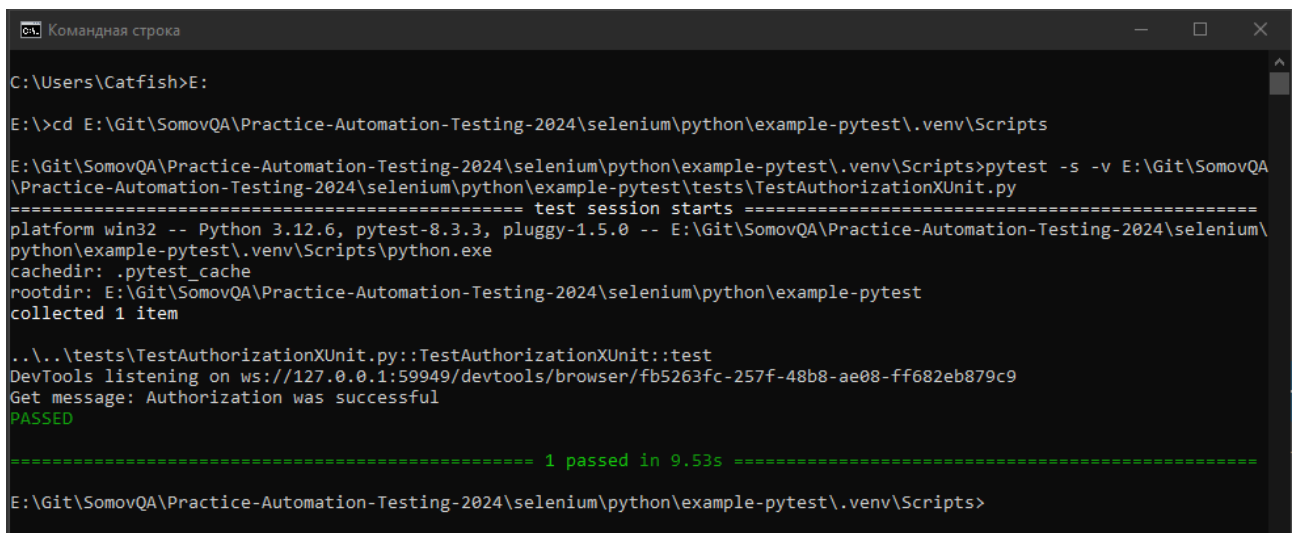


ИЛИ В КОНСОЛИ ВЫПОЛНИТЬ КОМАНДЫ

Е:

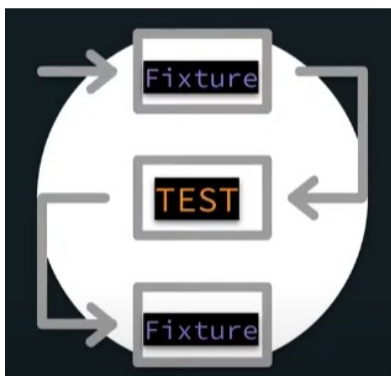
```
cd E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-pytest\.venv\Scripts
```

```
pytest -s -v E:\Git\SomovQA\Practice-Automation-Testing-2024\selenium\python\example-pytest\tests\TestAuthorizationXUnit.py
```



Практика PyTest с применением Fixture

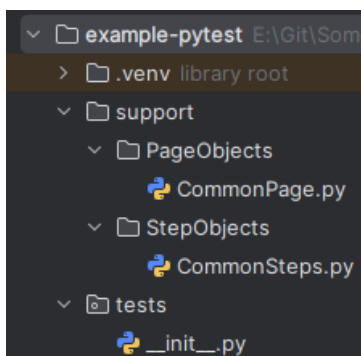
Фикстуры помогают сократить дублирующийся код



Порядок и частота вызова фикстур

1. session - один раз при запуске всех тестов
2. module - один раз в пакете
3. class - перед тестовым классом
4. function - перед каждым тестом

Описание паттернов PageObjects и StepObjects



В файле CommonPage.py

```
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait

class CommonPage:
    nameLogin = "login"
    namePassword = "pass"
    idButtonLogin = "buttonLogin"
    idResult = "result"
    idTextarea = "textarea"

    def getResultText(driver):
        element = driver.find_element(By.ID, CommonPage.idResult)
        wait = WebDriverWait(driver, timeout=5)
        wait.until(lambda d: element.is_displayed())
        text = driver.find_element(By.ID,
                                   CommonPage.idTextarea).get_property('value')
        print("Get message: " + text)
        return text
```

В файле CommonSteps.py

[illegible]

В файле TestAuthorizationFixture1.py описать автотест с использованием фикстур

```
import pytest

from selenium import webdriver
from support.PageObjects.CommonPage import CommonPage
from support.StepObjects.CommonSteps import CommonSteps

@pytest.fixture(scope="function")
def init_driver():
    driver = webdriver.Chrome()
    driver.maximize_window()
    return driver

def test(init_driver):
    tester = CommonSteps(init_driver)
    tester.driver.get('https://somovstudio.github.io/test_eng.html')
    tester.sendForm('admin', '0000')
    text = CommonPage.getResultText(tester.driver)
    assert text == 'Authorization was successful', "Получено некорректное сообщение"
    tester.driver.close()
    tester.driver.quit()
```

В файле TestAuthorizationFixture2.py использование фикстур как Setup, Test, Teardown

строка `yield driver` - останавливает выполнение функции `init_driver`, запускает тест `test_authorization` передает в него `driver` и после завершения тест возвращает управление в функцию `init_driver` где происходит закрытие браузера независимо от того успешно ли был пройден тест или нет.

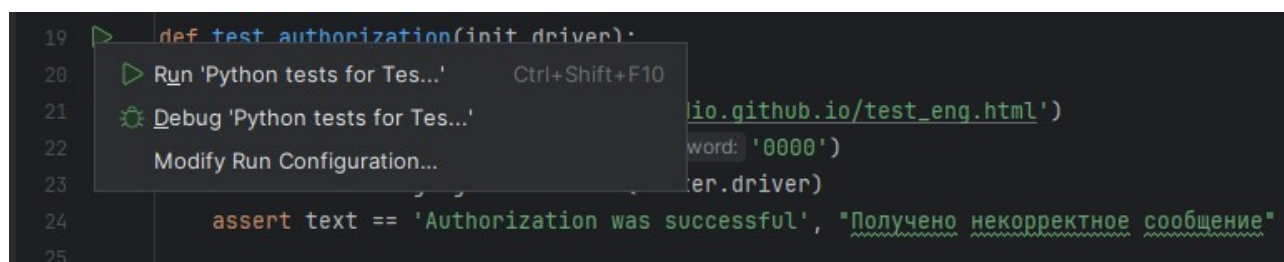
```
import pytest

from selenium import webdriver
from support.PageObjects.CommonPage import CommonPage
from support.StepObjects.CommonSteps import CommonSteps

@pytest.fixture(scope="session")
def init_driver():
    # Setup
    driver = webdriver.Chrome()
    driver.maximize_window()
    # Test
    yield driver
    # Teardown
    driver.close()
    driver.quit()

def test_authorization(init_driver):
    tester = CommonSteps(init_driver)
    tester.driver.get('https://somovstudio.github.io/test_eng.html')
    tester.sendForm('admin', '0000')
    text = CommonPage.getResultText(tester.driver)
    assert text == 'Authorization was successful', "Получено некорректное сообщение"
```

Автотест нужно запускать на функции `test_authorization`



Параметризация тестовых данных

Чтобы задать список данных для тестирования нужно использовать фикстуру `@pytest.mark.parametrize`

В файле `TestFixture.py` описать автотест следующим образом

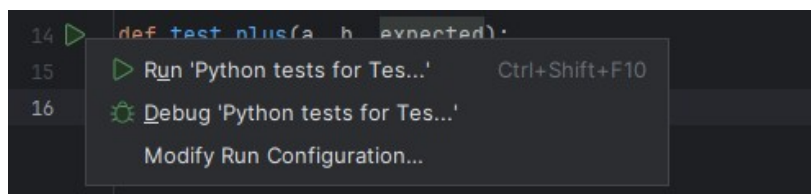
```
import pytest

def plus(a, b):
    return a + b

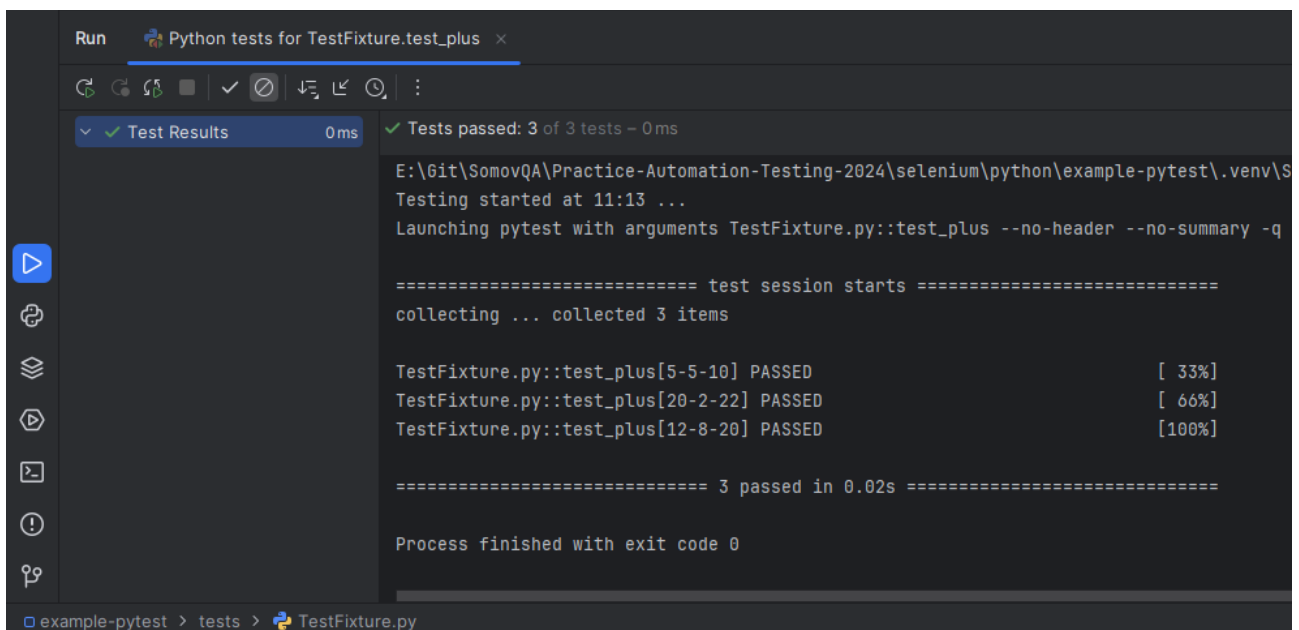
def minus(a, b):
    return a - b

@pytest.mark.parametrize("a, b, expected", [
    (5, 5, 10),
    (20, 2, 22),
    (12, 8, 20)
])
def test_plus(a, b, expected):
    actual = plus(a, b)
    assert expected == actual
```

Запуск автотеста



В результате тест будет запущен три раза под указанные параметры: a, b, expected



Полезные ссылки:

- Официальная документация Fixture
<https://docs.pytest.org/en/6.2.x/fixture.html>

Тестирование API с помощью Bruno

Скачать и установить [Wampserver](#) (сервер Apache, PHP, MySQL)

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");

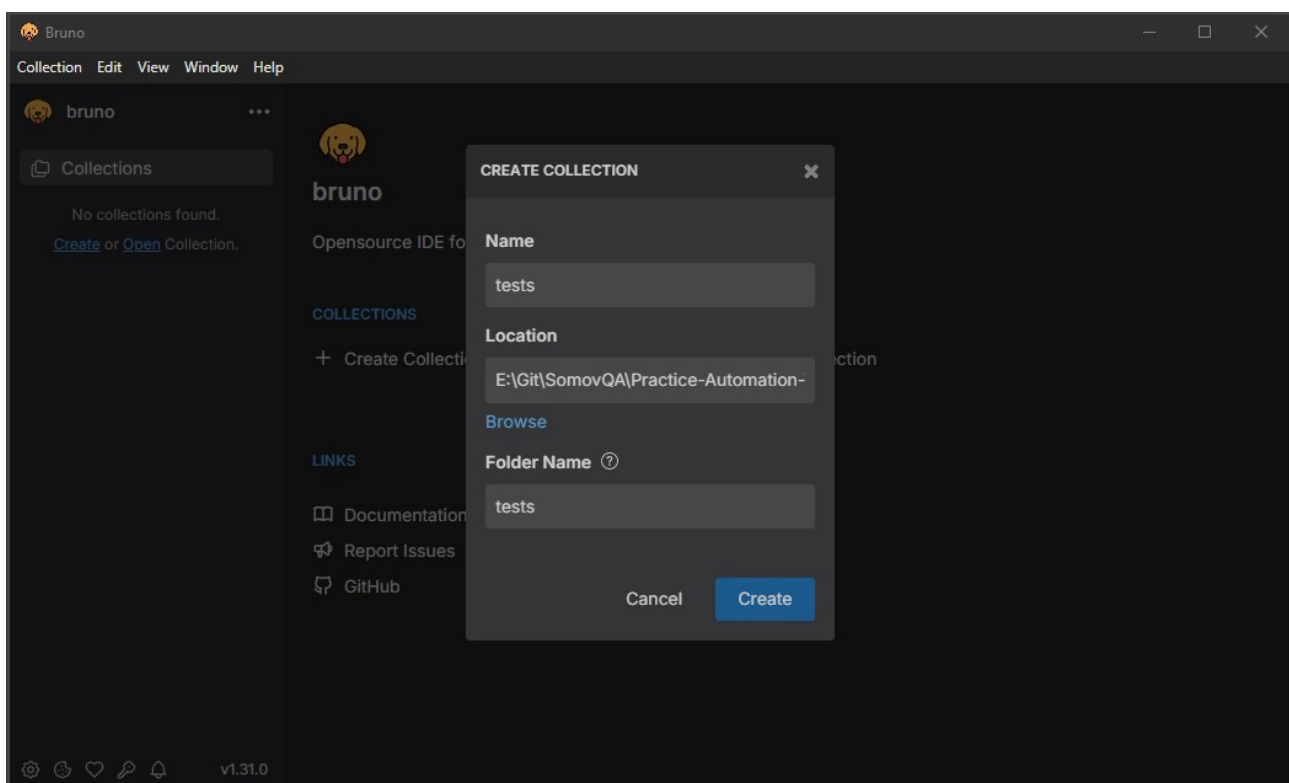
/*
    Пример обращения к этому API
    http://localhost/api/auth.php?name=admin&pass=0000
*/

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"}');
}
?>
```

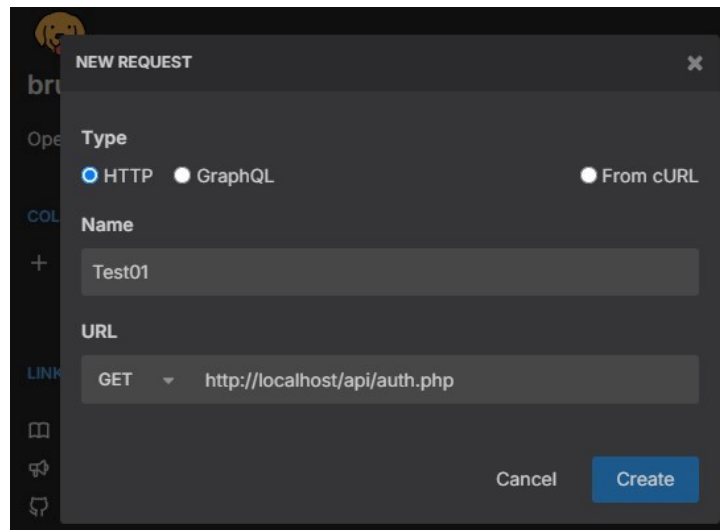
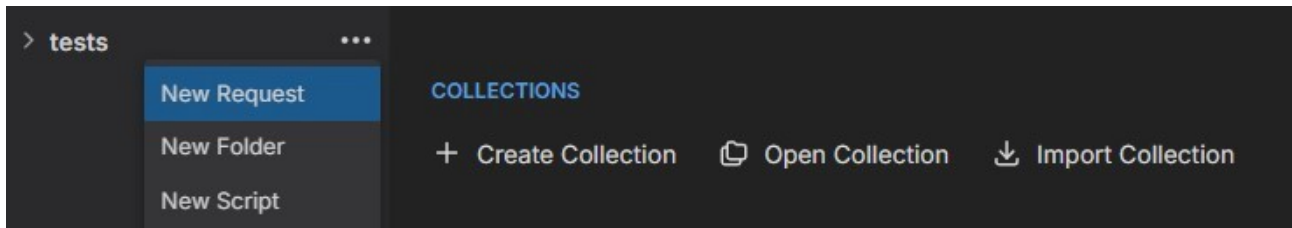
таким образом API будет по адресу <http://localhost/api/auth.php?name=admin&pass=0000>

Скачать и установить [Bruno](#)

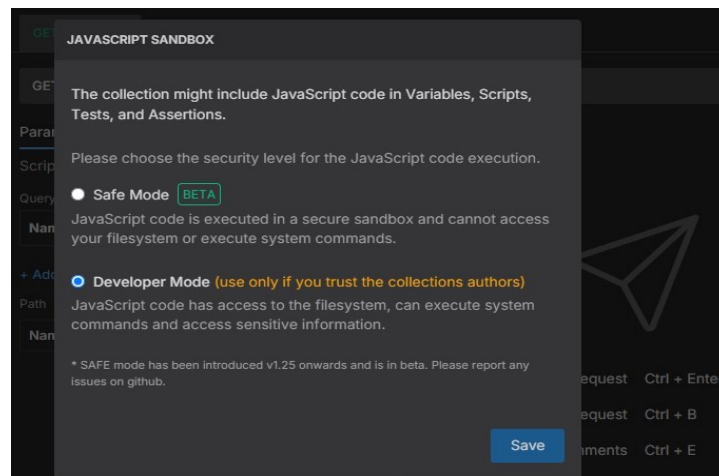
Запустить Bruno и создать коллекцию tests



В коллекции tests создать запрос Test01




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



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



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 }
```

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

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 });
```

Выполним запрос



POST Test02 x +

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

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

Response Headers⁸ Timeline Tests⁴

200 OK 3ms 82B

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 });

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.

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

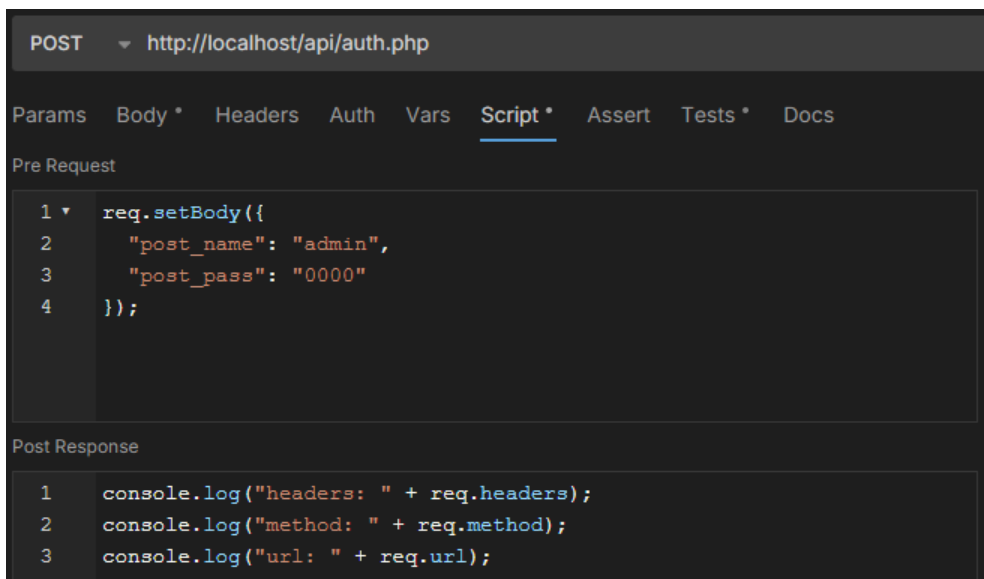
- Чтобы выполнить все запросы в папке, используйте:
bru run folder
- Если вам необходимо использовать определенную среду, вы можете передать ее с помощью параметра --env:
bru run folder --env Local
- Вы можете передавать переменные среды непосредственно в свою коллекцию с помощью параметра --env-var:
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`

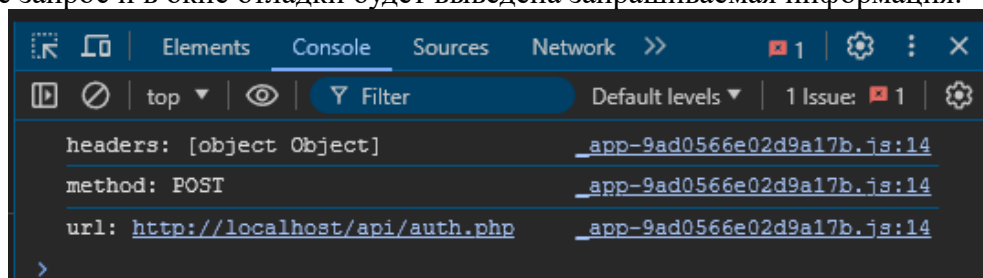
12. Отладка в окне Toggle Developer Tools

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>

