

8. Java_QA (Selenium, JUnit)

- Автоматизация тестирования, создание тестового фреймворка с использованием Selenium WebDriver
- Xpath, CSS локаторы (взаимодействие с элементами браузера)
- Работа с формами, перетаскивание объектов, загрузка и заливка файлов, ввод данных, сценарии авторизации
- Переходы между окнами браузера
- JUnit проверка сценариев

Внедрение зависимостей, pom.xml :

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
  </dependency>

  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.141.59</version>
  </dependency>

  <dependency>
    <groupId>io.github.bonigarcia</groupId>
    <artifactId>webdrivermanager</artifactId>
    <version>4.2.2</version>
  </dependency>

  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>2.0.0-alpha1</version>
  </dependency>
</dependencies>
```

Структура тестового класса, Case:

```
WebDriver driver;
WebDriverWait wait;
final String URL = "https://buggy-testingcup.pgs-soft.com/";
```

Метод инициализации, аннотация @Before, создание веб-драйвера, пример конфигурирования параметров браузера:

```
@Before
public void initialization() {
    HashMap<String, Object> chromePrefs = new HashMap<String, Object>();
    chromePrefs.put("profile.default_content_settings.popups", 0);
    chromePrefs.put("download.default_directory",
System.getProperty("user.dir"));
    ChromeOptions options = new ChromeOptions();
    options.setExperimentalOption("prefs", chromePrefs);
    WebDriverManager.chromedriver().setup();
    driver = new ChromeDriver(options);
    driver.manage().window().maximize();
    driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
    driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);
    driver.manage().window().maximize();
    wait = new WebDriverWait(driver, 10);
    driver.get(URL);
}
```

Метод завершения работы, аннотация @After, закрытие драйвера:

```
@After
public void tearDown() {
    if (driver != null) {
        driver.quit();
    }
}
```

Тест 1: Переход на страницу №1, ввод числа в форму, проверка корректного ввода, очистка поля и последующая проверка корректной очистки

```
@Test
public void inputNumberAndClearField() throws InterruptedException {
    selectTask("1");
    WebElement input = driver.findElement
        (By.xpath("//button[@data-product-name='Okulary']/../following-
sibling::input"));
    input.clear();
    sleep(2000);
    input.sendKeys("2");
    sleep(2000);
    driver.findElement(By.xpath("//button[@data-product-
name='Okulary']")).click();
    sleep(2000);
    int i = 1;
    WebElement sum = driver.findElement(By.className("summary-quantity"));
    Assert.assertEquals("Should Be Equal", "2", sum.getText());
    sleep(2000);
    driver.findElement(By.xpath("//button[.='Usun']")).click();
    Assert.assertEquals("Should Be Equal", "0", sum.getText());
}
```

Вспомогательный метод перехода на страницу с заданием, во избежание дублирования кода:

```
// Метод перехода на нужную страницу
private void selectTask(String num) {
    driver.findElement(By.xpath("//h2[.='Zadanie ' + num + '']")).click();
}
```

Результат:

Первичный ввод:

Twój koszyk		
Okulary (15.54)	2	Usuń
Łączna ilość produktów:		2
Do zapłaty:		31.08 zł

Последующая очистка поля:

Twój koszyk		
Łączna ilość produktów:		0
Do zapłaty:		0.00 zł

Тест 2: Переход на страницу №8. Выборка данных из выпадающего списка, по ключу, по индексу, по видимому тексту. Последующая проверка данных.

```
//Выборка в выпадающем списке по видимому тексту
@Test
public void checkSelectByVisibleText() {
    selectTask("8");
    Select select = new
Select(driver.findElement(By.id("task8_form_cardType")));
    select.selectByVisibleText("MasterCard");
    WebElement option = select.getFirstSelectedOption();
    Assert.assertEquals("MasterCard", option.getText());
    int i = 1;
}

//Выборка в выпадающем списке по индексу
@Test
public void checkSelectById() {
    selectTask("8");
    Select select = new
```

```

Select(driver.findElement(By.id("task8_form_cardType")));
select.selectByIndex(4);
WebElement option = select.getFirstSelectedOption();
Assert.assertEquals("Discover" , option.getText());
int i = 1;
}

//Выборка в выпадающем списке по ключу
@Test
public void checkSelectByValue() {
    selectTask("8");
    Select select = new
Select(driver.findElement(By.id("task8_form_cardType")));
select.selectByValue("dc");
WebElement option = select.getFirstSelectedOption();
Assert.assertEquals("Diners Club" , option.getText());
int i = 1;
}

```

Результат:

The screenshot shows a payment form with the following fields and a dropdown menu:

- Typ karty**: A dropdown menu currently showing "MasterCard". The menu is open, displaying a list of card types: American Express, American Express Corporate, Australian BankCard, Diners Club, Discover, JCB, MasterCard (highlighted in blue), Visa, Dankort (PBS), and Switch/Solo (Paymentech).
- Imię i nazwisko**: A text input field.
- Nr karty**: A text input field.
- Kod CVV**: A text input field with an information icon.
- Data ważności karty**: Two dropdown menus for "Miesiąc" (Month) and "Rok" (Year). The month is set to "January" and the year to "2021".
- Zapłać**: A blue button at the bottom right.

Тест 3: Переход на страницу №3, проверка, что поля ввода отключены, клик по кнопке активации полей и последующая проверка успешной активации

Метод проверки неактивности полей:

```

private void isElementDisabled(String element) {
    Assert.assertFalse(driver.findElement(By.id(element)).isEnabled());
}

```

Метод проверки активности полей:

```

private void isElementEnabled(String element) {
    Assert.assertTrue(driver.findElement(By.id(element)).isEnabled());
}

```

Метод активации полей при нажатии на соответствующую кнопку:

```
private void allowEdit() {
    Actions actions = new Actions(driver);

    actions.moveToElement(driver.findElement(By.className("caret"))).build().perform();
    actions.moveToElement(driver.findElement(By.className("caret-right"))).build().perform();
    driver.findElement(By.id("start-edit")).click();
}
```

Тестовый метод:

```
@Test
public void checkEnabledAndDisabledFields() {
    selectTask("3");
    isElementDisabled("in-name");
    isElementDisabled("in-surname");
    isElementDisabled("in-notes");
    isElementDisabled("in-phone");
    isElementDisabled("in-file");
    allowEdit();
    isElementEnabled("in-name");
    isElementEnabled("in-surname");
    isElementEnabled("in-notes");
    isElementEnabled("in-phone");
    isElementEnabled("in-file");
    int i = 1;
}
```

Результат:

Поля заблокированы:

Imię:	<input type="text" value="Salma"/>
Nazwisko:	<input type="text" value="Hayek"/>
Notatka:	<input type="text" value="Wprowadź notatkę"/>
Telefon:	<input type="text" value="19154175411"/>
Zdjęcie:	<input type="button" value="Wybierz plik"/>

Поля открыты для редактирования:

Imię:	<input type="text" value="Salma"/>
Nazwisko:	<input type="text" value="Hayek"/>
Notatka:	<input type="text" value="Wprowadź notatkę"/>
Telefon:	<input type="text" value="19154175411"/>
Zdjęcie:	<input type="button" value="Wybierz plik"/>
	<input type="button" value="Zapisz"/>

Тест 4: Переход на страницу №7, использование Drag&Drop, перетаскивание товара в корзину и последующая проверка введенного количества

```
@Test
public void addProductWithDragAndDrop() throws InterruptedException {
    selectTask("7");
    WebElement fieldProductNumber =
driver.findElement(By.xpath("//h4[.='Aparat']/following-
sibling::div/input"));
    fieldProductNumber.clear();
    fieldProductNumber.sendKeys("1");
    sleep(5000);
    WebElement productImage =
driver.findElement(By.xpath("//div[h4='Aparat']/preceding-
sibling::div/img"));
    WebElement basket = driver.findElement(By.cssSelector(".panel-body"));
    Actions action = new Actions(driver);
    action.dragAndDrop(productImage, basket).perform();
    sleep(5000);
    WebElement basketAmountForProduct =
driver.findElement(By.xpath("//span[@data-quantity-for='Aparat']"));
    Assert.assertEquals("1", basketAmountForProduct.getText());
}
```

Тест 5: Загрузка фото профиля и проверка корректного отображения, файл “123.png” лежит в корне проекта

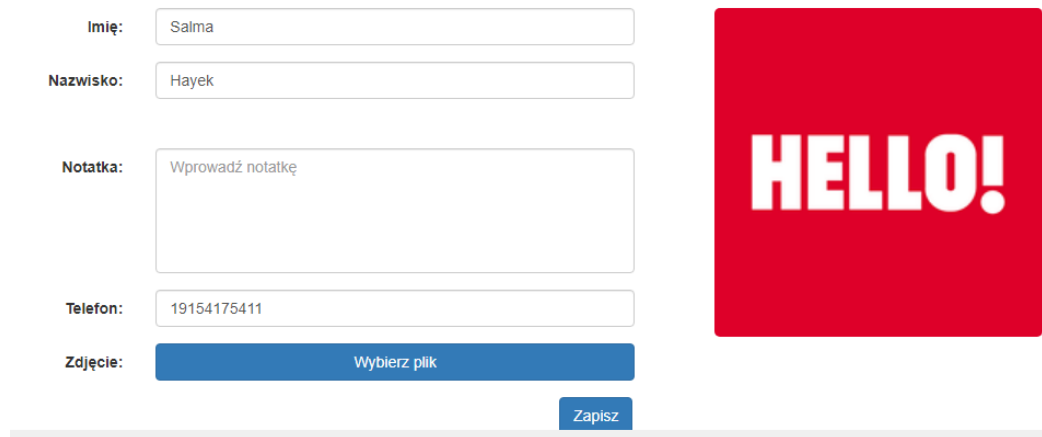
```
@Test
public void checkUploadFile() throws InterruptedException {
    selectTask("3");
    allowEdit();
    driver.findElement(By.id("in-file")).
        sendKeys(System.getProperty("user.dir") + "\\123.png");
    sleep(10000);
}
```

```

driver.findElement(By.id("save-btn")).click();
sleep(5000);
WebElement element = driver.findElement(By.cssSelector(".preview-
photo"));
Assert.assertTrue(element.isDisplayed());
int i = -1;
}

```

Результат:



Imię:

Nazwisko:

Notatka:

Telefon:

Zdjęcie:

HELLO!

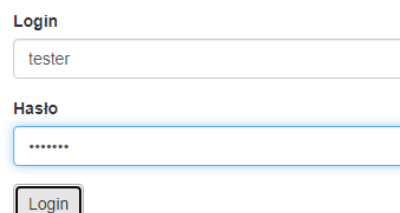
Тест 6: Авторизация, ввод логина, пароля, переход дальше по ссылке

```

@Test
public void isCorrectLogin() throws InterruptedException {
    selectTask("6");
    sleep(1000);
    driver.findElement(By.id("LoginForm__username")).sendKeys("tester");
    sleep(1000);
    driver.manage().timeouts().implicitlyWait(2, TimeUnit.SECONDS);
    driver.findElement(By.id("LoginForm__password")).sendKeys("123-xyz");
    sleep(1000);
    driver.findElement(By.id("LoginForm_save")).click();
    sleep(1000);
    driver.findElement(By.linkText("Pobierz plik")).click();
    sleep(1000);
    int i = 1;
}

```

Результат:



Login

Hasło

Тест 7: Переход по ссылке на файлообменник, скачивание файла и проверка соответствия

Метод проверки успешной загрузки файла:

```
private void isFileDownloaded(String file) {
    File folder = new File(System.getProperty("user.dir"));
    File[] listOfFiles = folder.listFiles();
    boolean found = false;
    File fl = null;
    for (File randomFile : listOfFiles) {
        if (randomFile.isFile()) {
            String fileName = randomFile.getName();
            if (fileName.matches(file)) {
                fl = new File(fileName);
                found = true;
            }
        }
    }
    Assert.assertTrue("File not found", found);
    fl.deleteOnExit();
}
```

Тестовый метод:

```
@Test
public void checkDownloadingFile() throws InterruptedException {
    driver.get("https://cloud.mail.ru/public/iH4E/gKhZ22N5v");
    sleep(5000);
    driver.findElement(By.className("ViewerToolbar__download--3R6Ry")).click();
    sleep(5000);
    isFileDownloaded("EYGGvYOERKSS.pdf");
}
```

Тест 8: Переключение между окнами браузера

Навигация по страницам с помощью итератора:

```
@Test
public void newWindowHandling() {
    driver.get("http://the-internet.herokuapp.com/windows");
    driver.findElement(By.linkText("Click Here")).click();
    waitForSecondWindow();
    Set<String> windows = driver.getWindowHandles();
    Iterator<String> itr = windows.iterator();
    String parentWindow = itr.next();
    String childWindow = itr.next();
    driver.switchTo().window(childWindow);
    Assert.assertEquals(driver.getTitle(), "New Window");
    driver.switchTo().window(parentWindow);
    Assert.assertEquals(driver.getTitle(), "The Internet");
}
```


Сохранение страниц в динамический список и навигация по ним:

```
@Test
public void newWindowHandlingWithList() {
    driver.get("http://the-internet.herokuapp.com/windows");
    driver.findElement(By.linkText("Click Here")).click();
    waitForSecondWindow();
    Set<String> windows = driver.getWindowHandles();
    List<String> winds = new ArrayList<>(windows);
    String parentWindow = winds.get(0);
    String childWindow = winds.get(1);
    driver.switchTo().window(childWindow);
    Assert.assertEquals(driver.getTitle(), "New Window");
    driver.switchTo().window(parentWindow);
    Assert.assertEquals(driver.getTitle(), "The Internet");
}
```

Запуск кейса: mvn clean test

Результат:

