УДК 004.415.53

Обновление стека инструментов для тестирования: причины и шаги перехода с Selenium на Selenide

Маркевич Даниил Владимирович[1] — магистрант кафедры «Информационные и вычислительные системы». E-mail: dmarkevich811@mail.ru

Хомоненко Анатолий Дмитриевич[1, 2] — доктор технических наук, профессор, профессор кафедры «Информационные и вычислительные системы» Петербургского государственного университета путей сообщения Императора Александра I, профессор кафедры «Математическое и программное обеспечение» Военно-космической академии имени А. Ф. Можайского. E-mail: khomonenko@pgups.ru

[1] Петербургский государственный университет путей сообщения Императора Александра I, Россия, Санкт-Петербург

[2] Военно-космическая академия имени А. Ф. Можайского, Россия, Санкт-Петербург

Для цитирования: Маркевич Д. В., Хомоненко А. Д. Обновление стека инструментов для тестирования: причины и шаги перехода с Selenium на Selenide // Интеллектуальные технологии на транспорте. 2024. № 2 (38). С. 57–68. DOI: 10.20295/2413-2527-2024-238-57-68

Аннотация. *Автоматизированное тестирование с использованием Selenium было стандартом в разработке ПО, но с ростом сложности приложений возникли потребности в более продвинутых инструментах, таких как Selenide. Рассматривается переход к Selenide, подчеркиваются расширенные возможности и удобство использования Selenide для автоматизированного тестирования.* ***Цель исследования:*** *демонстрация преимуществ перехода на Selenide для автоматизированного тестирования браузера, повышение стабильности тестирования и предоставление руководства для миграции.* ***Методы и средства:*** *включают настройку среды тестирования, перенос тестовых сценариев, оптимизацию и рефакторинг тестов. Используются такие функции Selenide, как автоматическое ожидание, сжатый синтаксис и улучшенная обработка ошибок. Приведены примеры и конфигурации ряда инструментов: Maven, Gradle и Allure. Исследование отражает процесс перехода на Selenide, демонстрируя улучшения в стабильности и удобочитаемости тестов. Приведены примеры тестовых сценариев, оптимизированных для повышения производительности и удобства обслуживания.* ***Практическая значимость:*** *заключается в повышении эффективности тестов. Рассмотрен комплексный процесс миграции, описаны этапы настройки, миграции сценариев и оптимизации, а также проблемы во время перехода и решения. Дальнейшие исследования целесообразно направить на оптимизацию производительности Selenide в крупномасштабных приложениях и изучение дополнительных функций.*

Ключевые слова: *автоматизированное тестирование, Selenium, Selenide, оптимизация тестов, Allure-отчеты.*

Введение

В отрасли разработки программного обеспечения всегда необходимо стремиться к развитию в соответствии с современными тенденциями. Развивающиеся и реализованные проекты становятся все сложнее, что означает необходимость разрабатывать более надежные и эффективные тесты. Selenium — одна из самых популярных платформ для тестирования веб-приложений,

уже несколько лет является стандартом в отрасли. С появлением новых технологий возникли новые инструменты, которые обеспечивают расширенную функциональность и удобный пользовательский интерфейс [1].

В этой статье рассмотрен один из них — Selenide, являющийся надстройкой над Selenium и представляющий собой надежную платформу для упрощенного написания автоматизированных тестов. Легко сопровождающиеся тесты, интуитивно понятный API, встроенный механизм ожидания и поддержка динамически изменяющихся элементов позволяют Selenide стать достойной альтернативой традиционному тестированию на Selenium [2].

В последующих разделах рассмотрено, какие преимущества существуют при переходе с Selenium на Selenide. Также представлено подробное сравнение двух инструментов и пошаговое руководство, которое помогает существенно упростить процесс проверки на создание более стабильных и поддерживаемых тестов.

Причины перехода на Selenide в автоматизации тестирования

Selenide разработан в 2011 году для обеспечения более простого и понятного синтаксиса по сравнению с Selenium. Недостатки Selenium, такие как написание большого количества шаблонного кода, ручное управление ожиданиями и взаимодействием с браузером, могли быть трудоемкими для тестировщиков, именно поэтому, для того чтобы избавить пользователя от этих проблем, создавался Selenide, который позволяет упрощать процесс написания и поддержки автоматизированных тестов [3].

Для полного осознания преимуществ Selenide необходимо иметь понимание фундаментальных отличий между Selenium и Selenide. Несмотря на то что обе платформы имеют одну и ту же главную цель (автоматизированное вебтестирование), предложенные ими подходы и функциональные возможности значительно отличаются [4].

Первым отличием Selenide является синтаксис и дизайн API. Хотя API Selenium эффективен, он иногда может быть громоздким и повторяющимся. Для выполнения стандартного теста в Selenium необходимо написать ряд строк кода, которые определяют местоположение элементов на странице, выполняют действия с ними и проверяют условия. В отличие от предшественника Selenide обладает более кратким и адаптивным интерфейсом программирования, что позволяет уменьшить объем кода, необходимого для решения аналогичных задач. Примеры с поиском элемента и выполнением действия с помощью Selenium и Selenide представлены на рис. 1 и 2.

```
WebDriver driver = new ChromeDriver();
driver.get("http://example.com");
WebElement element = driver.findElement(By.id("someId"));
element.click();
```

Рис. 1. Реализация поиска элемента с последующим щелчком через Selenium

```
open( relativeOrAbsoluteUrl: "http://example.com");
$(id("someId")).click();
```

Рис. 2. Реализация поиска элемента с последующим щелчком через Selenide

Следующий отличительный аспект — встроенные механизмы ожидания. При проведении тестирования веб-сайтов часто возникают сложности с обработкой динамического контента и асинхронных событий. Если настройка явных или неявных ожиданий в Selenium будет выполнена неправильно, это может привести к сбоям в тестировании. Selenide же имеет встроенные механизмы ожидания, которые автоматически справляются со временем загрузки элементов и предотвращают возникновение ошибок при взаимодействии с ними [5].

На рис. 3 и 4 приведены примеры использования Selenium и Selenide, где происходит ожидание отображения элемента с последующим щелчком.

Далее стоит рассмотреть удобство чтения и сопровождения тестов, так как благодаря гибкому API Selenide тестовые сценарии становятся более понятными и выразительными. Это особенно важно при работе с большим количеством тестов по мере развития проекта. В связи с этим тестировщики могут легко понимать и изменять тесты, без необходимости разбираться в сложном коде [6].

Пример с подтверждением наличия ожидаемого текста с помощью Selenium и Selenide представлен на рис. 5 и 6.

Также важным отличительным фактором является настройка веб-драйвера и браузера. Время на установку и обслуживание может быть выше из-за необходимости тестировщиков Selenium контролировать настройки WebDriver и браузера. Selenide решает эту проблему, так как предоставляет умные настройки по умолчанию и автоматическое управление конфигурацией браузера [7].

Рис. 7 и 8 демонстрируют примеры настройки WebDriver с использованием библиотек Selenium и Selenide. После анализа основных различий между Selenium и Selenide отметим конкретные преимущества, благодаря которым использование Selenide становится значительно более предпочтительным вариантом для автоматизированного тестирования [8].

Основное преимущество Selenide заключается в его упрощенном синтаксисе, который заметно сокращает количество обычного кода для написания тестов.

```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
WebElement element = wait.until(ExpectedConditions
        .visibilityOfElementLocated(By.id("someId")));
element.click();
```

Рис. 3. Реализация ожидания отображения элемента с щелчком через Selenium

```
open( relativeOrAbsoluteUrl: "http://example.com");
$(id("someId")).click();
```

Рис. 4. Реализация ожидания отображения элемента с щелчком через Selenide

```
WebElement element = driver.findElement(By.id("someId"));
String text = element.getText();
assertEquals( expected: "Expected Text", text);
```

Рис. 5. Реализация проверки наличия ожидаемого текста через Selenium

```
$(id("someId")).shouldHave(text("Expected Text"));
```

Рис. 6. Реализация проверки наличия ожидаемого текста через Selenide

```
ChromeOptions options = new ChromeOptions();
options.addArguments("--headless=new");
WebDriver driver = new ChromeDriver(options);
driver.get("http://example.com");
```

Рис. 7. Реализация настройки WebDriver через Selenium

```
Configuration.headless = true;
open( relativeOrAbsoluteUrl: "http://example.com");
```

Рис. 8. Реализация настройки WebDriver через Selenide

Для долгосрочного успеха любого проекта при автоматизации тестирования является важным иметь код, который возможно прочитать и поддерживать. Использование Selenide повышает понятность тестовых сценариев, делая их более интуитивными и простыми для чтения. На рис. 9 показано, как осуществляется отправка формы через Selenide.

Далее рассмотрим такое преимущество, как автоматическая обработка ожиданий и синхронизация. В Selenium частой проблемой является обработка динамического контента и синхронизация. Для обработки этих сценариев Selenium требует применения явного или неявного ожидания, что может вызывать ошибки и некорректное выполнение теста. Selenide легко решает данную проблему благодаря своим встроенным механизмам ожидания. Selenide автоматически ожидает каждого взаимодействия, чтобы элемент достиг требуемого состояния, прежде чем продолжить. Это обеспечивает стабильность и надежность тестов.

Также Selenide обеспечивает повышенную стабильность во время тестирования благодаря использованию встроенного режима ожидания и улучшенной обработке асинхронных событий. Большое значение имеет эта стабильность для крупных проектов, которые содержат сложные веб-приложения. В таких случаях проблемы с синхронизацией и динамическим контентом могут часто нарушить результаты тестов. При уменьшении вероятности ложных результатов тестов Selenide повышает надежность результатов выполнения всего набора тестирования [9].

Кроме того, Selenide обеспечивает дополнительные возможности для отчетов об ошибках и отладки, которые включают подробную информацию о сбоях тестирования. Если тест Selenide обнаруживает сбой, то он автоматически создает скриншот экрана и регистрирует состояние браузера на момент сбоя. Это может быть весьма полезным для определения причины возникновения. Отчеты об ошибках улучшают эффективность процесса тестирования путем быстрого выявления и устранения проблем со стороны тестировщиков [10]. Для настройки генерации скриншотов при сбое можно использовать всего одну команду, которая представлена на рис. 10.

```
$(id("text field element")).setValue("Example Text");
$(id("submit button element")).click();
```

Рис. 9. Реализация отправки формы через Selenide

```
Configuration.reportsFolder = "target/reports";
```

Рис. 10. Команда для генерации скриншотов в указанную директорию при сбое

Отметим, что Selenide с легкостью интегрируется с JUnit, TestNG и CI/CD-конвейерами для разработки и тестирования. С помощью этой интеграции команды могут легко внедрить Selenide в свои существующие рабочие процессы без проблем. Более того, благодаря совместимости Selenide со множеством популярных платформ разработки он становится универсальным решением для широкого спектра проектов [11].

Тем не менее у некоторых тестировщиков могут возникнуть опасения относительно перехода с Selenium, несмотря на преимущества Selenide.

1. Процесс обучения: Selenide использует нетрудный синтаксис и подход, так что тестировщики, знакомые с Selenium, легко адаптируются к ним. API Selenide является интуитивно понятным и легким для изучения благодаря своей конструкции, и многие специалисты считают, что изучение Selenide быстро окупится благодаря увеличению производительности и сокращению расходов на обслуживание [12].

2. Совместимость с имеющимися тестами Selenium: тестировщикам стоит обратить внимание на совместимость уже имеющихся тестов на Selenium с Selenide. Selenide позволяет постепенно преобразовывать существующий код Selenium без необходимости полного переписывания для Selenide, так как он создан поверх Selenium. Этот пошаговый подход позволяет каждой команде выполнять миграцию в собственном темпе, минимизируя возможные сбои при проведении тестирования. Перевод на Selenide не представляет сложности, и пример его реализации в одном методе показан на рис. 11 и 12.

3. Производительность: некоторые специалисты напрасно сомневаются в том, что функции Selenide (ожидания и автоматическое управление браузером) оказывают влияние на производительность. На практике использование Selenide для оптимизации часто приводит к более быстрому выполнению тестов, поскольку требуется меньше ручной обработки ожидания и сокращаются проблемы с синхронизацией. Также использование поддержки параллельного тестирования в Selenide может значительно увеличить производительность и делать его особенно подходящим для обработки больших наборов тестов [13].

4. Гибкость и индивидуальная настройка: Selenium обладает гибкостью, которая позволяет тестировщикам в значительной степени настраивать свои сценарии проверки. Selenide поддерживает эту гибкость, предлагая разумные настройки по умолчанию и встроенные функции, которые покрывают наиболее распространенные возможности использования. Если это необходимо, тестировщики все еще могут использовать базовое Selenium API для работы с уникальными или сложными сценариями тестирования, так как Selenide не имеет ограничений в этом отношении [14].

```
@Step("Выбор раздела 'Научные конференции'")
public MainPage scientificConferencesClick() {
    webDriverWait.until(ExpectedConditions.elementToBeClickable(scienceModule)).click();
    webDriverWait.until(ExpectedConditions.elementToBeClickable(scientificConferences)).click();
    return this;
}
```

Рис. 11. Существующий код Selenium для открытия раздела «Научные конференции»

```
@Step("Выбор раздела 'Научные конференции'")
public MainPage scientificConferencesClick() {
    $x(scienceModule).click();
    $x(scientificConferences).click();
    return this;
}
```

Рис. 12. Открытие раздела «Научные конференции» в результате перехода на Selenide

Наконец, Selenide предлагает привлекательную альтернативу Selenium, которая решает множество проблем и ограничений традиционных платформ веб-тестирования. Его упрощенный синтаксис, автоматическая обработка ожидания и повышенная стабильность делают его привлекательным выбором для команд, которые хотят улучшить свои возможности в автоматизированном тестировании. Selenide помогает тестировщикам сосредоточиться на гарантировании качества и надежности своих приложений, сокращая сложность написания и поддержки тестов [15].

Шаги перехода от Selenium к Selenide

Чтобы гарантировать бесперебойность и эффективность процесса, при переходе с Selenium на Selenide требуется провести несколько конкретных этапов. Главная цель этих этапов состоит в том, чтобы использовать расширенные функции Selenide с сохранением надежности и стабильности тестирования. В этом разделе будет приведен подробный анализ каждого шага с обозначением конкретных примеров и рекомендаций, которые помогут облегчить переход [16].

Для использования Selenide следует настроить рабочую среду, чтобы она была способна поддерживать новую платформу. Далее необходимо произвести обновление зависимостей проекта, настроить интегрированную среду разработки (IDE) и убедиться в полной совместимости уже используемых инструментов и инфраструктуры [17].

Прежде чем использовать Selenide, необходимо внести зависимости в проект. Если разработка осуществляется на Maven, необходимо прописать следующую зависимость в файле pom.xml (рис. 13).

В случае использования Gradle необходимо добавить зависимость в файл build, приведенную на рис. 14.

Также следует убедиться, что настройки в среде разработки IDE позволяют правильно распознавать библиотеку Selenide. При обновлении конфигурации проекта множество современных интегрированных сред разработки (IDE), например, IntelliJ IDEA и Eclipse, автоматически загружают и настраивают все необходимые зависимости [18].

Кроме того, необходимо проверить интеграцию Selenide с уже существующими инструментами и инфраструктурой. Это включает CI/CD-конвейеры, инструменты для создания отчетов о тестировании и все пользовательские утилиты, которые могли быть уже внедрены в проект.

После настройки среды следующим шагом является начало миграции уже имеющихся тестовых сценариев Selenium в Selenide. Для достижения правильной и эффективной работы требуется редактирование тестовых сценариев при использовании API-функций и Selenide [19].

Для знакомства с функциями и синтаксисом Selenide рекомендуется начать переносить несложные тестовые сценарии. В стандартных простых тестах обычно осуществляются проверки базовых действий, включая переход на страницу, нажатие кнопок и сравнение текста. Выполнение переноса простого теста иллюстрируется на рис. 15 и 16.

```
<dependency>
    <groupId>com.codeborne</groupId>
    <artifactId>selenide</artifactId>
    <version>7.3.1</version>
    <scope>test</scope>
</dependency>
```

Рис. 13. Зависимость Selenide для проекта на Maven

```
dependencies {
  testImplementation 'com.codeborne:selenide:7.3.1'
}
```

Рис. 14. Зависимости Selenide для проекта на Gradle

```java
@BeforeEach
public void getDriver() {
    driver = MANAGER.getDriver();
}

@DisplayName("Скачивание документа с планом научных событий")
@ParameterizedTest
@MethodSource("selenium.junit.PgupsTestData#test7TestData")
public void downloadingDocumentWithPlanOfScientificEvents(String expectedUrl) {
    WebDriver driver = MANAGER.getDriver();
    driver.manage().window().maximize();
    driver.get("https://www.pgups.ru/");

    WebDriverWait webDriverWait = new WebDriverWait(MANAGER.getDriver(), Duration.ofSeconds(10));

    webDriverWait.until(ExpectedConditions.visibilityOfElementLocated(centralBanner));
    webDriverWait.until(ExpectedConditions.elementToBeClickable(scienceModule)).click();
    webDriverWait.until(ExpectedConditions.elementToBeClickable(scientificConferences)).click();

    webDriverWait.until(ExpectedConditions.visibilityOfElementLocated(pageTitle));
    WebElement report = webDriverWait.until(ExpectedConditions.visibilityOfElementLocated(planOfScientificEvents));
    MANAGER.getDriver().navigate().to(report.getAttribute( name: "href"));
    assertEquals(MANAGER.getDriver().getCurrentUrl(), expectedUrl);
}

@AfterEach
public void quitTest() {
    driver.quit();
}
```

Рис. 15. Тест на скачивание документа с планом научных событий через Selenium

```java
@DisplayName("Скачивание документа с планом научных событий")
@ParameterizedTest
@MethodSource("selenide.junit.PgupsTestData#test7TestData")
public void downloadingDocumentWithPlanOfScientificEvents(String fileName) {
    open( relativeOrAbsoluteUrl: "https://www.pgups.ru/");

    $x(centralBanner).should(visible, Duration.ofSeconds(20));
    $x(scienceModule).click();
    $x(scientificConferences).click();

    $x(pageTitle).should(visible, Duration.ofSeconds(20));
    File report = $x(planOfScientificEvents).download();
    checkIfFileExist(fileName);
}
```

Рис. 16. Тест на скачивание документа с планом научных событий на Selenide

После миграции тестовых примеров необходимо приступить к оптимизации и рефакторингу, чтобы максимально эффективно использовать функциональные возможности Selenide. В рамках этого шага улучшается читаемость тестов, снижается количество дублирования кода и обеспечивается стабильность во время тестирования, так как Selenide API предоставляет возможность разрабатывать собственные команды и утилиты, которые помогут упростить повседневные операции, и для повышения надежности тестирования начать пользоваться функциями ожидания и обработки ошибок из библиотеки [20].

Затем все тесты должны быть тщательно проверены на надежность и способность обрабатывать динамический контент, задержку в сети и другие распространенные проблемы, которые могут повлиять на стабильность теста за счет интеграции тестов Selenide в конвейер CI/CD, где происходит автоматическое выполнение тестов при каждой сборке. Это обеспечит постоянную связь и раннее обнаружение любых потенциальных проблем. Также стоит использовать инструменты Selenide для создания отчетов о тестировании и анализа результатов. Allure или TestNG предлагает подробные отчеты и информацию о выполнении тестов, которые помогают обнаруживать и устранять проблемы. Для создания отчетов Allure следует обратить внимание на зависимость, которая показана на рис. 17.

Для обеспечения эффективности формирования тестовой системы рекомендуется систематически выполнять тесты с применением Selenide и сразу же реагировать на какие-либо сбои. Проверку результатов тестов и мониторинга можно осуществить различными способами, например через использование интерфейса отчетов allure. Пример такого отчета изображен на рис. 18.

```
<dependency>
    <groupId>io.qameta.allure</groupId>
    <artifactId>allure-junit5</artifactId>
    <version>2.27.0</version>
</dependency>
```

Рис. 17. Зависимость для генерации allure-отчетов

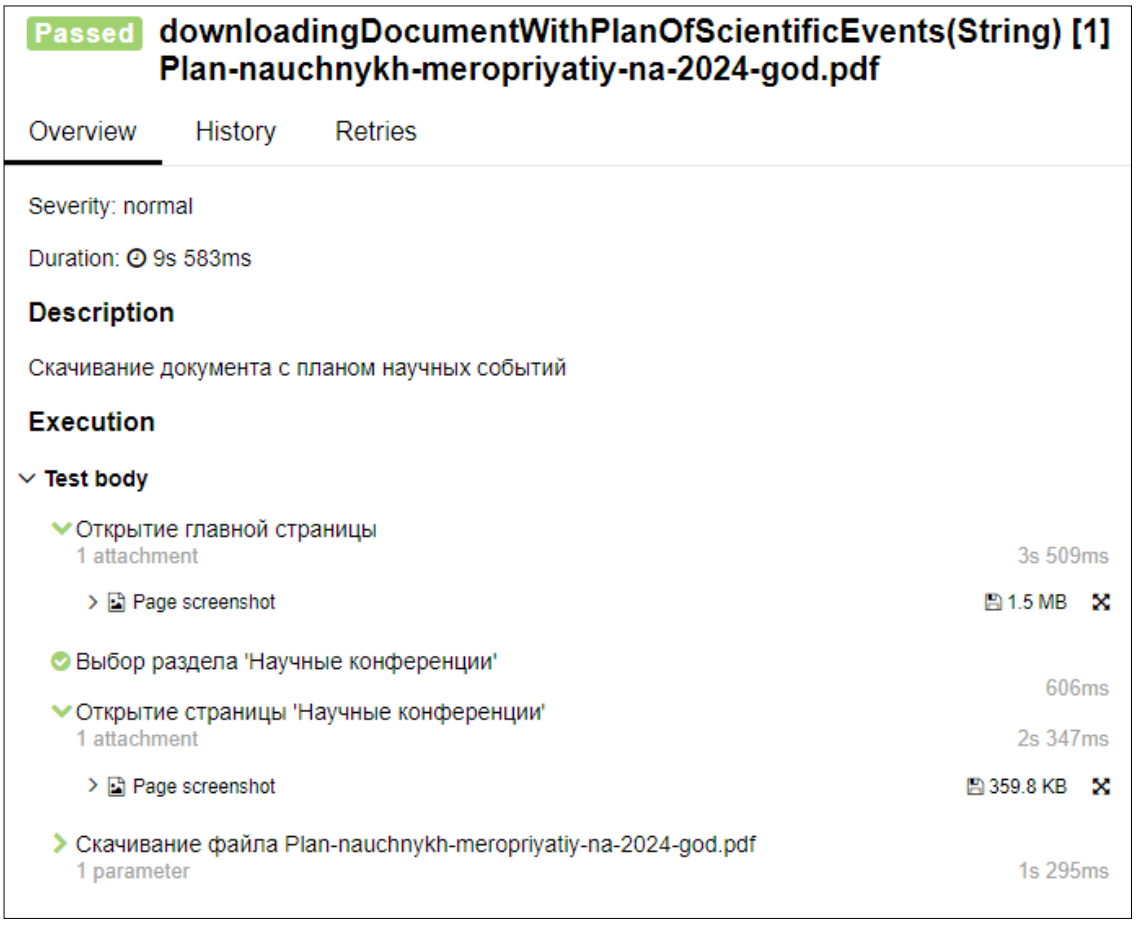

Рис. 18. Allure-отчет с результатами пройденного теста

Таким образом, переход к Selenide включает ряд хорошо определенных шагов, которые обеспечивают плавный и эффективный процесс миграции. Путем настройки среды, миграции тестовых скриптов, оптимизации и рефакторинга тестов и интеграции с конвейерами CI/CD можно использовать улучшенные возможности Selenide для улучшения стабильности, читабельности и поддерживаемости тестовой системы. С тщательным планированием и непрерывным развитием переход к Selenide позволит обеспечивать разработку программного обеспечения высокого качества с большей эффективностью [21].

Заключение

Переход от Selenium к Selenide является стратегическим шагом, который может значительно усилить качество автоматизированного тестирования. Перейдя на Selenide, можно получить доступ к более пользовательскому API, встроенным механизмам ожидания и улучшенной обработке ошибок, что поспособствует созданию более стабильных и поддерживаемых тестовых скриптов. В ходе проделанной работы очерчены критические шаги, необходимые для совершения этого перехода, — от настройки среды и миграции тестовых скриптов до оптимизации и интеграции с конвейерами CI/CD [22].

Тщательно планируя и выполняя каждый шаг, можно обеспечить плавный и эффективный переход к Selenide. Преимущества этого перехода станут очевидны по мере того, как тестовая система станет более прочной и легкой в обслуживании, что итоге приведет к более высокому качеству программного обеспечения и более быстрым циклам разработки.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Косов Е. С., Попов А. М. Разработка системы автоматизированного функционального тестирования интернет-магазина Брэндмэйкер // Пищевые инновации и биотехнологии. 2022. С. 30–31.

2. Букреева И. Р., Муртазина А. Р. Создание автоматизированных сценариев тестирования web-приложения на примере сайта «Магазин для творчества» // Инновационное развитие техники и технологий в промышленности (ИНТЕКС-2021). С. 61–63.

3. Петкун В. О. Применение типовых элементов при автоматизированном тестировании. БНТУ. 2022. С. 161–164.

4. Петрова А. И. Исследование методов и средств автоматизированного тестирования web-приложений // Новые информационные технологии в научных исследованиях (НИТ-2021). 2021. С. 119–121.

5. Архипов И. С. Внедрение автоматизированного тестирования в agile-разработке // Universum: технические науки. 2023. С. 25–30.

6. Gojare S., Joshi R., Gaigaware D. Analysis and design of selenium webdriver automation testing framework // Procedia Computer Science. 2015. Vol. 50. P. 341–346.

7. Яницкая Т. С., Моренов И. Р. Обзор и анализ существующих паттернов проектирования автоматизированных фреймворков тестирования API. МЦНП «Новая наука». 2023. С. 45–53.

8. Ramya P., Sindhura V., Sagar P. V. Testing using Selenium web driver // 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT). IEEE, 2017. P. 1–7.

9. Кириллов С. С. Внедрение системы управления тестовыми данными в проект по автоматизации тестирования // Международный журнал гуманитарных и естественных наук. 2022. С. 150–152.

10. Маркевич Д. В., Хомоненко А. Д., Ермаков С. Г. От Foxpro к PostgreSQL: оптимизация, эффективное управление данными и генерация отчетов // Наукоемкие технологии в космических исследованиях Земли. 2024. № 1. С. 21–30.

11. Сударчиков Г. Е. Анализ инструментов для проведения автоматизированного функционального тестирования программного обеспечения // Проблемы развития современного общества. 2024. С. 129–132.

12. Васильев В. А. Автоматизация процесса тестирования информационных систем за счет разработки специализированных программных инструментов // Студенческая молодежь XXI века: наука, творчество, карьера, цифровизация. 2022. С. 63–72.

13. Галаган Т. А., Греков П. А. Проектирование системы автоматизированного тестирования задач по олимпиадному программированию // Вестник Амурского государственного университета, серия «Естественные и экономические науки». 2021. С. 42–45.

14. Биджиев М. Х., Ковалева К. А. Автоматизация тестирования при разработке ПО: инструменты и подходы // Актуальные проблемы научных исследований: теоретические и практические аспекты. 2023. С. 73–79.

15. Буравов А. А., Дузбаев Н. Т. Использование контейнеризации для автоматизированного тестирования программного обеспечения в онлайн-образовании // Universum: технические науки. 2022. С. 56–60.

16. Глухов К. А., Зарубин И. Б. Особенности формирования модульных и интеграционных тестов при разработке современных информационных систем. КОГРАФ-2022. 2022. С. 7–8.

17. Альтшулер И. О. Selenium WebDriver как инструмент функционального тестирования веб-приложений. Витебск: ВГУ имени П. М. Машерова. 2021. С. 286–288.

18. Смольский С. С. Роль, назначение и проблемы автоматизированного тестирования. БГУИР. 2022. С. 121–123.

19. Петренко С. А., Петренко А. А. Цифровая платформа тестирования и верификации программного кода на основе автомата динамического контроля // Дистанционные образовательные технологии. 2020. С. 400–405.

20. Кириллов С. С. Внедрение автоматизированных тестов в систему непрерывной интеграции // Научные вести. 2022. С. 37–44.

21. Бугаенко Р. С. Автоматизированное тестирование при разработке программного обеспечения // Международная научно-техническая конференция молодых ученых. 2020. С. 3835–3840.

22. Маркевич Д. В., Харланова В. В., Хомоненко А. Д. Интеграция систем бизнес-аналитики с системами управления базами данных на транспорте // Наукоемкие технологии в космических исследованиях Земли. 2023. Т. 15, № 2. С. 41–48.

Дата поступления: 03.06. 2024
Решение о публикации: 03.06.2024

Updating the Stack of Testing Tools: Reasons and Steps for Switching from Selenium to Selenide

Daniil V. Markevich[1] — Master's student at the Department of Information and Computing systems of Emperor Alexander I St. Petersburg state transport university. E-mail: dmarkevich811@mail.ru

Anatoly D. Khomonenko[1, 2] — Doctor of Technical Sciences, Full Professor, Professor of the Department of Information and Computing systems of Emperor Alexander I St. Petersburg state transport university, Professor of the Department "Mathematics and Software" in VKA named after A. F. Mozhaisky. E-mail: khomonenko@pgups.ru

[1] Emperor Alexander I Petersburg State Transport University, Saint Petersburg, Russia
[2] A. F. Mozhaysky Military Space Academy, Saint Petersburg, Russia

For citation: Markevich D. V., Khomonenko A.D. Updating the stack of testing tools: reasons and steps for switching from Selenium to Selenide // Intelligent technologies on transport. 2024. No. 2 (38). P. 57–68. (In Russian). DOI:10.20295/2413-2527-2024-238-57-68

Abstract. *Automated testing using Selenium was the standard in software development, but with the increasing complexity of applications, there was a need for more advanced tools such as Selenide. The transition to Selenide is considered, the advanced features and convenience of using Selenide for automated testing are emphasized. The purpose of the study is to demonstrate the benefits of switching to Selenide for automated browser testing, increase the stability of testing and provide guidance for migration.* ***Methods and tools.*** *These include setting up the testing environment, porting test scenarios, optimizing and refactoring tests. Selenide features such as automatic waiting, compressed syntax, and improved error handling are used. Examples and configurations of a number of tools are given: Maven, Gradle and Allure. The study reflects the process of switching to Selenide, demonstrating improvements in the stability and readability of texts. Examples of test scenarios optimized to increase productivity and ease of maintenance are given.* ***Practical significance.*** *It is to increase the effectiveness of tests. The complex migration process is considered, the stages of setup, scenario migration and optimization, as well as problems during the transition, and solutions are described. Further research should be directed to optimizing Selenide performance in large-scale applications and exploring additional features.*

Keywords: *automated testing, Selenium, Selenide, optimization of test, Allure reports.*

REFERENCES

1. Kosov E. S., Popov A. M. Razrabotka sistemy avtomatizirovannogo funkcional'nogo testirovaniya internet-magazina Brendmejker // Pishchevye innovacii i biotekhnologii. 2022. S. 30–31. (In Russian)

2. Bukreeva I. R., Murtazina A. R. Sozdanie avtomatizirovannyh scenariev testirovaniya web-prilozheniya na primere sajta “Magazin dlya tvorchestva” // Innovacionnoe razvitie tekhniki i tekhnologij v promyshlennosti (INTEKS-2021). S. 61–63. (In Russian)

3. Petkun V. O. Primenenie tipovyh elementov pri avtomatizirovannom testirovanii. BNTU. 2022. S. 161–164. (In Russian)

4. Petrova A. I. Issledovanie metodov i sredstv avtomatizirovannogo testirovaniya web-prilozhenij // Novye informacionnye tekhnologii v nauchnyh issledovaniyah (NIT-2021). 2021. S. 119–121. (In Russian)

5. Arhipov I. S. Vnedrenie avtomatizirovannogo testirovaniya v agile-razrabotke // Universum: tekhnicheskie nauki. 2023. S. 25–30. (In Russian)

6. Gojare S., Joshi R., Gaigaware D. Analysis and design of selenium webdriver automation testing framework // Procedia Computer Science. 2015. T. 50. S. 341–346.

7. Yanickaya T. S., Morenov I. R. Obzor i analiz sushchestvuyushchih patternov proektirovaniya avtomatizirovannyh frejmvorkov testirovaniya API. MCNP “Novaya nauka”. 2023. S. 45–53. (In Russian)

8. Ramya P., Sindhura V., Sagar P. V. Testing using Selenium web driver // 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT). IEEE, 2017. P. 1–7.

9. Kirillov S. S. Vnedrenie sistemy upravleniya testovymi dannymi v proekt po avtomatizacii testirovaniya // Mezhdunarodnyj zhurnal gumanitarnyh i estestvennyh nauk. 2022. S. 150–152. (In Russian)

10. Markevich D. V., Homonenko A. D., Ermakov S. G. Ot Foxpro k PostgreSQL: optimizaciya, effektivnoe upravlenie dannymi i generaciya otchetov // Naukoemkie tekhnologii v kosmicheskih issledovaniyah Zemli. 2024. № 1. S. 21–30. (In Russian)

11. Sudarchikov G. E. Analiz instrumentov dlya provedeniya avtomatizirovannogo funkcional'nogo testirovaniya programmnogo obespecheniya // Problemy razvitiya sovremennogo obshchestva. 2024. S. 129–132. (In Russian)

12. Vasil'ev V. A. Avtomatizaciya processa testirovaniya informacionnyh sistem za schet razrabotki specializirovannyh programmnyh instrumentov // Studencheskaya molodezh' XXI veka: nauka, tvorchestvo, kar'era, cifrovizaciya. 2022. S. 63–72. (In Russian)

13. Galagan T. A., Grekov P. A. Proektirovanie sistemy avtomatizirovannogo testirovaniya zadach po olimpiadnomu programmirovaniyu // Vestnik Amurskogo gosudarstvennogo universiteta, seriya “Estestvennye i ekonomicheskie nauki”. 2021. S. 42–45. (In Russian)

14. Bidzhiev M. H., Kovaleva K. A. Avtomatizaciya testirovaniya pri razrabotke PO: instrumenty i podhody // Aktual'nye problemy nauchnyh issledovanij: teoreticheskie i prakticheskie aspekty. 2023. S. 73–79. (In Russian)

15. Buravov A. A., Duzbaev N. T. Ispol'zovanie kontejnerizacii dlya avtomatizirovannogo testirovaniya programmnogo obespecheniya v onlajn-obrazovanii // Universum: tekhnicheskie nauki. 2022. S. 56–60. (In Russian)

16. Gluhov K. A., Zarubin I. B. Osobennosti formirovaniya modul'nyh i integracionnyh testov pri razrabotke sovremennyh informacionnyh sistem. KOGRAF-2022. 2022. S. 7–8. (In Russian)

17. Al'tshuler I. O. Selenium WebDriver kak instrument funkcional'nogo testirovaniya veb-prilozhenij. Vitebsk: VGU imeni P. M. Masherova. 2021. S. 286–288. (In Russian)

18. Smol'skij S. S. Rol', naznachenie i problemy avtomatizirovannogo testirovaniya. BGUIR. 2022. S. 121–123. (In Russian)

19. Petrenko S. A., Petrenko A. A. Cifrovaya platforma testirovaniya i verifikacii programmnogo koda na osnove avtomata dinamicheskogo kontrolya // Distancionnye obrazovatel'nye tekhnologii. 2020. S. 400–405. (In Russian)

20. Kirillov S. S. Vnedrenie avtomatizirovannyh testov v sistemu nepreryvnoj integracii // Nauchnye vesti. 2022. S. 37–44. (In Russian)

21. Bugaenko R. S. Avtomatizirovannoe testirovanie pri razrabotke programmnogo obespecheniya // Mezhdunarodnaya nauchno-tekhnicheskaya konferenciya molodyh uchenyh. 2020. S. 3835–3840. (In Russian)

22. Markevich D. V., Harlanova V. V., Homonenko A. D. Integraciya sistem biznes-analitiki s sistemami upravleniya bazami dannyh na transporte // Naukoemkie tekhnologii v kosmicheskih issledovaniyah Zemli. 2023. T. 15. № 2. S. 41–48. (In Russian)

Received: 03.06.2024
Accepted: 03.06.2024