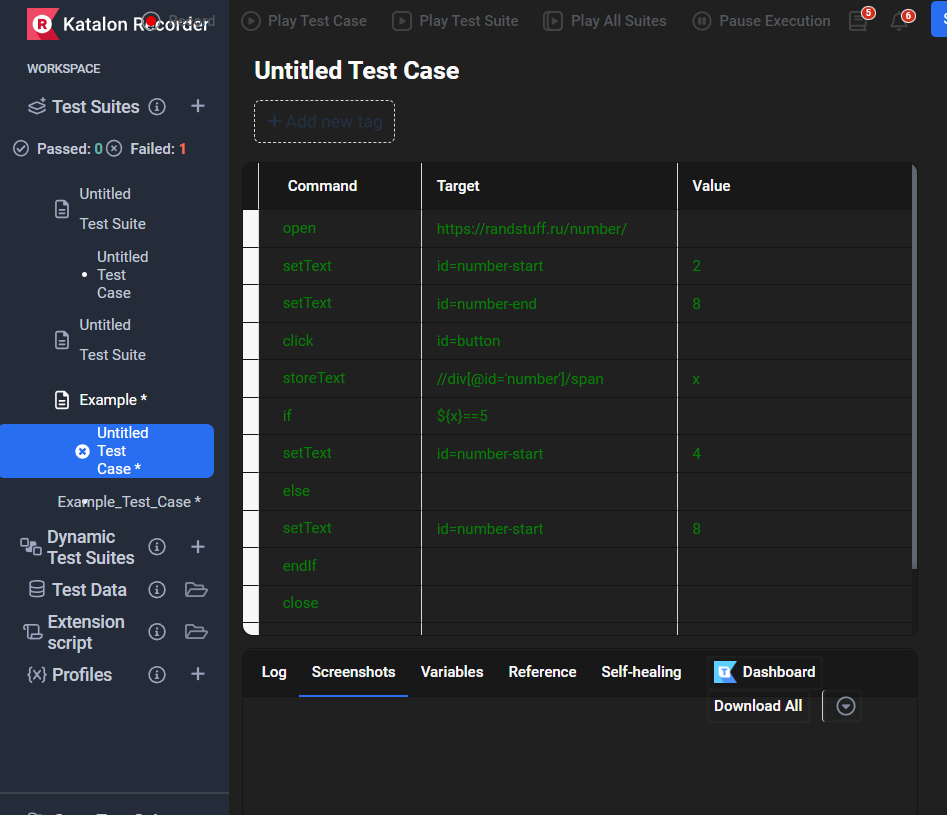
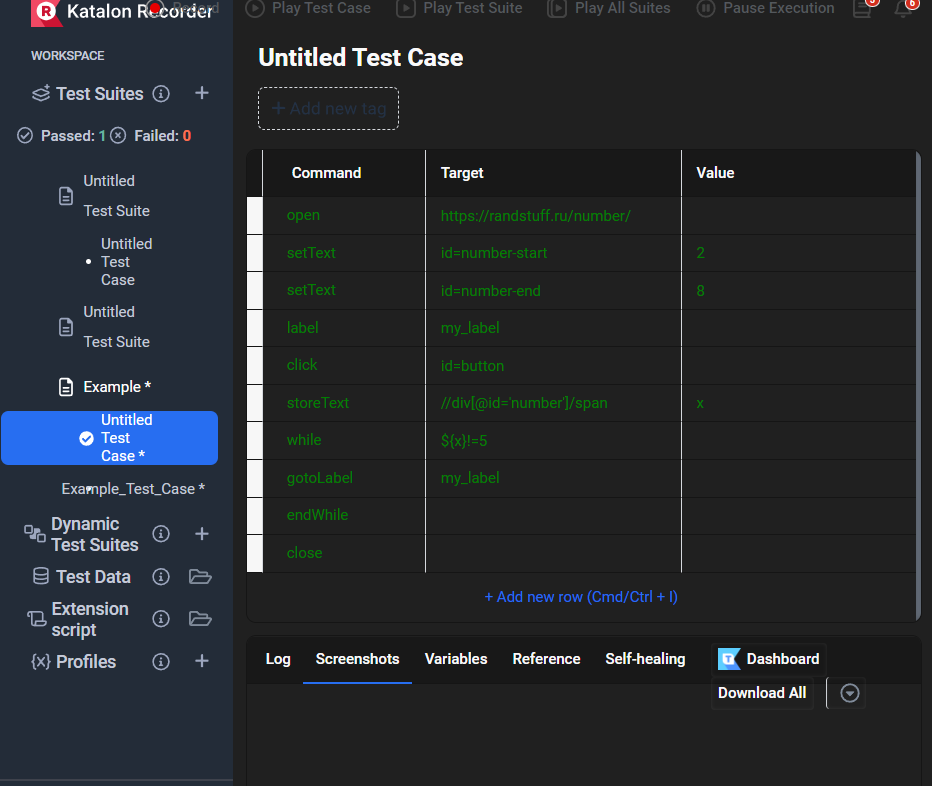
**Лабораторная работа №3***“Разработка тестов на основе технологии Record&Playback в Katalon Recorder”*

**Цель:** разработать тесты на основе технологии Record&Playback в Katalon Recorder.

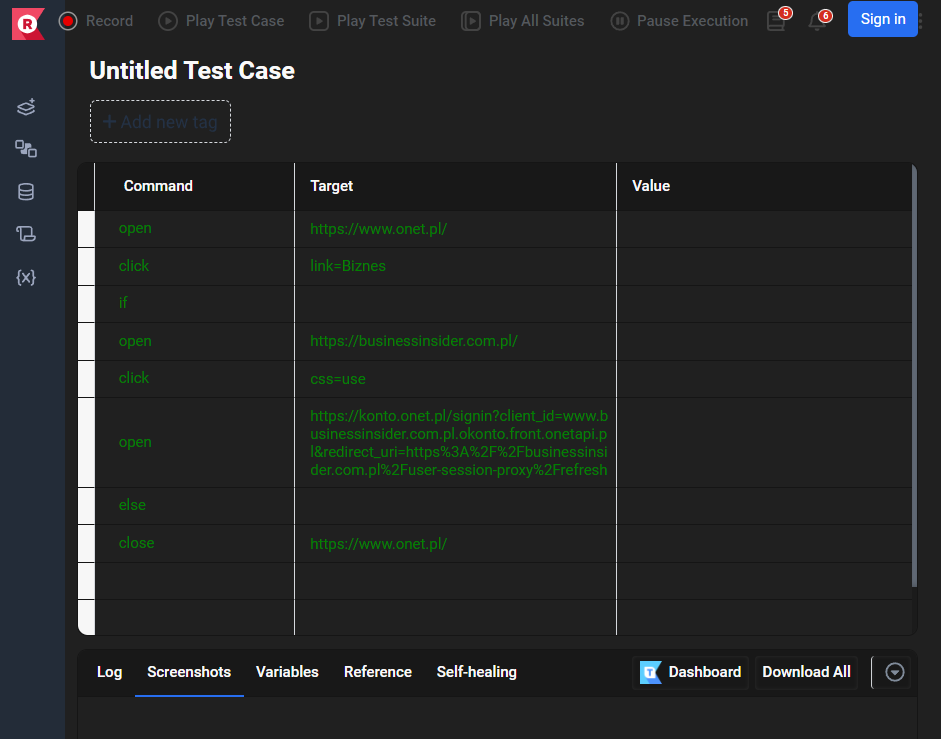
**Пример:**

****

****

**Задание 1:**

**No10. Создать скрипт с использованием if-конструкции для тестирования onet.pl.**

****

**Контрольные вопросы:**

1. Что такое Katalon Recorder?

Katalon Recorder – инструмент, который был разработан как альтернатива Selenium.

2. Основные функции Katalon Recorder?

Основные функции:

1. С помощью данного инструмента вы можете захватить необходимые элементы веб-приложения и записать выполняемые с ними действия. Вы можете создавать новые тестовые случаи, редактировать существующие, а также воспроизводить автоматические тестовые случаи. Katalon Recorder позволяет выполнять все эти действия легко и быстро.

2. Все поддерживаемые Selenese функции доступны в Katalon Recorder

3. Тестовые сценарии могут быть импортированы в Katalon Recorder (в скриптовом режиме) и экспортированы для приведенных ниже языков программирования и фреймворков.

4. Katalon Recorder поддерживается последними версиями Firefox и Chrome.

5. Данный инструмент работает на ядре Selenium 3.

6. Несколько тестовых случаев могут быть объединены в один тестовый набор. Такие объединенные тестовые случаи будут выполняться в виде полного тестового набора.

7. Тестовые случаи могут быть импортированы в Katalon Studio – полностью бесплатный инструмент для автоматизации тестирования, поддерживающий key-driven- и data-driven-подходы, TDD/BDD-тестирование, тестирование API, а также включающий такие возможности как: применение паттерна Page Object Model, управление и запись процесса тестирования, написание расширенных скриптов, параллельное выполнение, выполнение CI-интеграции и многие другие возможности!

8. Katalon Recorder позволяет записывать и воспроизводить автоматизированные тестовые сценарии не только пользователям Katalon Studio, но и тем, кто пользуется устаревшей версией Selenium IDE и другими популярными фреймворками с открытым исходным кодом.

3. Детальное описание основных функций Katalon Recorder.

1. Панель инструментов. Панель инструментов состоит из 6 кнопок: «New» (Создать), «Record» (Запись), «Play» (Воспроизвести), «Export» (Экспортировать), «Help» (Подсказка) и «Settings» (Настройки).

2. Test Case Explorer.

3. Окно тестовых случаев.

4. Вкладки «Журнал», «Справка», «Параметры». Недостатки Katalon Recorder:

1. Существует ряд команд, которые на данный момент находятся на стадии разработки. Например: sendKeys является экспериментальной командой. Вероятно, в будущем эта команда будет заменена на другую – typeKeys. Таким образом, есть вероятность того, что при использовании экспериментальных команд придется обновлять некоторые тестовые сценарии.

2. В инструменте отсутствует функция отображения базового URL-адреса. В Selenium IDE такая функция была очень удобна для инициализации тестовых случаев в нескольких разных доменах.

3. Кроме описанных выше недостатков в инструмент также есть и другие ошибки. В разделах «Suggestions» (Предложения) и «Katalon Automation Recorder Bugs» (Журнал ошибок Katalon Automation) вы можете найти информацию о существующих ошибках и способах их устранения.

4. Недостатки Katalon Recorder.

1. Есть определенные команды, которые на данный момент являются экспериментальными. Например, команда 'sendKeys' является таковой. ...

2. Опция указания базового URL-адреса также не представлена нигде в инструменте. ...

3. В Firefox на панели инструментов Katalon Recorder отсутствует кнопка «Settings».

5. Недостатки технологии Record&Playback.

Основная идея крайне проста. При первом прогоне теста мы записываем ответы сторонних систем, а при последующих воспроизводим записанные ранее ответы. Давайте посмотрим, какие из вышеозвученных проблем мы тогда решим.

Во-первых, эти тесты долго отрабатывают только в первый раз, когда вы пишете кассеты. Кассеты — это как раз те файлы, куда записывается трек ответов от сторонних систем. Но при последующих прогонах все выполняется со скоростью сравнимой с юнит-тестами, потому-что запросы не уходят никуда дальше локалхоста и, при этом, мы не тратим время на поднятие зависимостей. А поскольку чаще всего запись кассеты единичная операция, а вот прогон тестов в пайплайне — частая, получается, что тесты отрабатывают быстро когда это нужнее всего. Благодаря этому они легко интегрируются в пайплайн, ведь внешних зависимостей нет, а значит ваша Jenkins-джоба может работать также, как если бы это были юнит-тесты. И они не flaky поскольку никакие флуктуации сетевой связности или работы сложных внешних компонентов не оказывают на них влияния.

Но при этом для них не требуется писать много кода, как для юнит-тестов, потому-что проблема мокирования зависимостей теперь снимается с ваших плеч. Вы можете использовать ваше уже имеющееся окружение для записи кассет. Например, упомянутый ранее QA-environment.

И по объему проверяемой логики эти тесты превосходят юнит также, как и интеграционные, потому-что по сути они такие и есть. Чтобы записать кассету нужно, чтобы ваши запросы прошли все валидации в компонентах-зависимостях вашего приложения.

Вам определенно понадобится кое-что сделать с вашим приложением, чтобы запросы уходили не во внешнюю сеть, а в библиотеку, которая обеспечит эту логику: первый прогон пишем, потом воспроизводим. Но обычно для сетевого взаимодействия мы используем какую-то библиотеку, поэтому слишком сильной связности с кодом эти тесты иметь не будут. Замена одной библиотеки — это не мокирование каждого второго класса в коде.

Наконец, эти тесты говорят о работе системы даже больше, чем традиционные, потому-что в кассетах у вас будут записаны те детали взаимодействия с вашими зависимостями, которые в обычном случае вы могли бы посмотреть только под отладчиком, запустив интеграционный тест.

Решена проблема поддержания моков в актуальном состоянии, потому-что всегда можно просто стереть кассету и просто выполнить тест на запись снова. Ну а поскольку сервисы реальные, то никаких проблем с упрощением в моках мы тоже не испытываем.

Можно было бы удивиться, почему при всей простоте идеи, это не используется повсеместно. Но я думаю, что объяснение этому все-таки есть. Чисто умозрительно сразу возникают определенные сомнения, относительно того, будет ли это так просто, как звучит на словах. Возможно как раз какое-то из этих сомнений как раз сейчас пришло вам на ум и вы сидите и думаете: «Не, ну это не будет так просто, потому-что...» Вот на эти «потому-что» я сейчас и попытаюсь ответить. По крайней мере на те, о которых мне известно. Может в комментариях вы обогатите мою копилку новыми «потому-что».