


2. Selenium Webdriver (Yandex Market)

Задание	Task 2. Selenium Webdriver (yandex market)
Цель задания	Изучение и применение на практике Selenium Webdriver
Язык	Java, Python, C#
Тип задания	Автоматизация тестирования
Необходимые компетенции и техническая подготовка	Коммуникация; Знание ООП; Знание Web-технологий
Время на выполнение задания	14 ч

Описание задания

Необходимо разработать автотест для следующего тест-кейса:

#	Step	Expected Result
1	Зайти на страницу https://market.yandex.ru	Главная страница Yandex.Market открылась
2	Выполнить процесс авторизации	Вход успешно выполнен. Пользователь авторизован
3	Получить список популярных категорий товаров	
4	Перейти на случайную категорию из списка	Открыта страница выбранной категории товаров
5	Перейти на главную страницу сайта	
6	Кликнуть на «Все категории», получить список всех категорий. Все найденные категории записать в csv файл, который должен создаваться тестом автоматически	csv файл со списком всех категорий товаров
7	Проверить, что в списке «Всех категорий», полученном на шаге 6, содержатся все категории популярных товаров, полученные в шаге 3	Список «Все категории» включает в себя список «Популярные категории»
8	Выполнить log out	Пользователь успешно вышел из учетной записи

Требования к выполнению задания	<ol style="list-style-type: none"> 1. Работа с элементами UI в решении должна быть реализована на основании паттерна PageObject. 2. Организация механизма создания экземпляра браузера, а также дальнейшей работы с ним должна быть реализована по шаблону Singleton. 3. Организация автотестов (выполнение проверок), а также непосредственный запуск должен быть реализован при помощи механизмов библиотеки TestNG/PyTest/JUnit/MSTest. 4. Проект должен собираться автоматически. Зависимости/используемый библиотеки должны быть указаны в pom(Java)/requirements.txt(Python)/NuGet dependencies (C#) файлах. 5. Вынести настройки общие настройки проекта (имя браузера, url сайта, путь к директории) в отдельный конфигурационный файл (xml json yaml and etc.) и разработать класс, который будет читать оттуда данные. 6. Тест должен работать в двух браузерах (Firefox, Chrome). 7. Для конфигурации запуска тестов на разных браузерах в целях сокращения дублирующегося кода должна быть реализована Browser Factory (на основе шаблона проектирования Factory Method) 8. Тест должен поддерживать запуск в браузерах на OS Windows (Win 10) и Linux (Ubuntu). 9. В тесте должен использоваться хотя бы 1 раз локатор/селектор каждого из типов: <ul style="list-style-type: none"> • By.XPath • By.CSS 10. В тесте должны быть использованы методы click, sendKeys, getText/Text, getAttribute. 11. В тесте должен быть по крайней мере 1 клик через Actions.Click. 12. В задании должна быть выполнена работа с ожиданиями: fluent wait, implicitly wait, explicitly wait. 13. Код теста должен соответствовать принятым в компании Coding Guidelines 14. Код задания необходимо поместить в отдельную ветку (Branch) репозитория студента. Branch name = t2_selenium-webdriver-ym.
Ожидаемый результат выполнения задания	<ol style="list-style-type: none"> 1. Создан Pull/Merge Request из ветки с заданием в master 2. Письмо ментору, содержащее: <ul style="list-style-type: none"> • Ссылку на Pull/Merge Request кода задания в master • Для Java/Python: консольная команда для запуска тестов (через maven / pytest)Консольная команда для запуска тестов (через maven / pytest) • Время, затраченное на выполнение задания • На что было затрачено время сверх оценки времени на задание, если было превышение • Что вызвало наибольшие сложности при выполнении задания • Какие остались нерешенные проблемы, если остались • Тема письма = TCXXX. <studentName (e.g. v.pupkin)>. Task 2: Selenium Webdriver Results