OTUS

Специальная цена

Java QA Engineer

Углубленная автоматизация тестирования на стеке Java

Длительность курса: 104 академических часа

1 WebDriver

1	**Главное про автоматизацию тестирования**	обозначить уровни автоматизированного тестирования и соответствующие правила; определить критерии качества автотестов; проанализировать принципы тестирования; научиться создавать проект с минимальным набором обязательных атрибутов; создать проект с минимальным набором обязательных атрибутов.
2	**Настраиваем окружение, пишем первый тест**	объяснить, как выбрать тестовый фреймворк; настроить логирование, интеграцию с GIT и параметризированный запуск тестов. Домашние задания 1 Создать новый maven-проект для автоматизации тестирования и залить его на GitHub Цель: В результате выполнения дз вы создадите новый maven-проект для автоматизации тестирования и залить его на GitHub 1 Откройте IDE 2 Создайте новый проект (maven) 3 Настройте для проекта файл .gitignore (https://www.gitignore.io/) 4 В файле pom.xml укажите зависимости для -- Selenium Java (https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java)

-- WebDriverManager (https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager)
-- jUnit (https://mvnrepository.com/artifact/junit/junit/4.12)
5 создайте репозиторий на github и залейте в master код проекта
6 создайте новую ветку в репозитории и в рамках нее создайте тест, который:
-- с помощью WebDriverManager, настраивает ChromeDriver
-- открывает в браузере Chrome страницу https://otus.ru/
— проверяет title страницы
- логирует шаги теста с помощью log4j
7 сделайте push кода с тестом в новую ветку
8 сделайте pull request в мастер из новой ветки, убедитесь, что не возникло конфликтов и код можно мержить

Домашнее задание принимается в виде ссылки на GitHub репозиторий
Срок сдачи: 48 часов до следующего занятия.

Подглядывать можно сюда:
https://habr.com/ru/company/otus/blog/452908/
Использовать owner http://owner.aeonbits.org/docs/usage/

---

3 **Локаторы**

проанализировать все типы локаторов;
научиться строить сложные локаторы;
проанализировать какие возникают ошибки при работе с элементами;
рассмотреть подходы к хранению локаторов.

---

4 **Конфигурация драйвера при старте, desired capabilities, параметры браузеров, работа с cookies, настройка ожиданий**

научиться конфигурировать драйвер при старте, настраивать его под специфичные задачи;
научиться задавать настройки, специфичные для конкретного браузера.

Домашние задания

1 Реализовать WebDriverFactory класс

Цель: В результате выполнения дз, участники создадут для себя Factory-класс, предоставляющй удобный интерфейс для запуска драйверов разных браузеров

Создайте класс WebDriverFactory со статическим методом create();

Метод create() принимает обязательный параметр webDriverName и необязтельный параметр options, а возвращает соответствующий имени вебдрайвер с заданными (если были) options

Примеры использования
WebDriver wd =
WebDriverFactory.createNewDriver("chrome");
или
FirefoxOptions options = new FirefoxOptions();
WebDriver wd = WebDriverFactory.createNewDriver("firefox", options);

Про паттерн Factory: https://refactoring.guru/ru/design-patterns/factory-method

---

5 **Команды чтения и управления состоянием элементов. Свойства элементов html-страницы, получение специфичных свойств элементов**

проанализировать какие свойства элементов страницы доступны для чтения средствами автоматизации и какие проверки можно на них строить;
объяснить какие элементы Selenium считает interactable и clickable.

---

6 **Ожидания в Selenium. Работа с явными и неявными ожиданиями, сравнение подходов. Знакомство с Expected Conditions**

применять явные и неявные ожидания для повышения стабильности тестов и осуществления сложных проверок.

Домашние задания

1 Написать автотест для каталога Яндекс.Маркет

Цель: В результате выполнения дз вы реализуете автоматический тест, используя Java + Selenium.

Шаги теста:
- Открыть в Chrome сайт Яндекс.Маркет - "Электроника"-> "Смартфоны"
- Отфильтровать список товаров: Samsung и Xiaomi
- Отсортировать список товаров по цене (от меньшей к большей)
- Добавить первый в списке Samsung
-- Проверить, что отобразилась плашка "Товар {имя товара} добавлен к сравнению"
- Добавить первый в списке Xiaomi
-- Проверить, что отобразилась плашка "Товар {имя товара} добавлен к сравнению"
- Перейти в раздел Сравнение
-- Проверить, что в списке товаров 2 позиции

Домашнее задание принимается в виде ссылки на GitHub репозиторий
Срок сдачи: 48 часов до следующего занятия.

---

7 **Работа с нативными окнами браузера: Alert, Prompt, Confirm, iFrame, Tabs, BasicAuth**

поработать с нативными элементами браузера.

---

8 **Upload files, executing JavaScript**

объяснить, как решать нестандартные задачи с помощью выполнения JavaScript на странице.

2 Architecture

1	**Архитектура проекта**	применить паттерны проектирования при разработке проекта (Strategy, Proxy, Builder, Singleton, etc).
2	**Page object**	проанализировать теорию паттерна и его применение. Домашние задания 1 Реализуйте автоматический тест, используя Java + Selenium + POM Цель: В результате выполнения дз вы реализуете автоматический тест, используя Java + Selenium + POM. Шаги теста: - Открыть https://otus.ru - Авторизоваться на сайте - Войти в личный кабинет - В разделе "О себе" заполнить все поля "Личные данные" и добавить не менее двух контактов - Нажать сохранить - Открыть https://otus.ru в "чистом браузере" - Авторизоваться на сайе - Войти в личный кабинет - Проверить, что в разделе "О себе" отображаются указанные ранее данные Домашнее задание принимается в виде ссылки на GitHub репозиторий Срок сдачи: 48 часов до следующего занятия.
3	**Page factory, ScreenPlay**	использовать паттерны, альтернативные "стандартному" PageObject.
4	**Dependency Injection. Google Guice, PicoContainer, Spring**	проанализировать преимущества использования DI в разработке, а также внедрить его в свой проект.
5	**Contract testing with Spring**	

3 BDD

1 **Подход BDD**

объяснить суть подхода и основные инструменты для написания и имплементации сценариев.

---

2 **Архитектура проекта, использующего BDD**

внедрить BDD-подход в существующий проект и посмотреть, как архитектурно меняется проект.

Домашние задания

1 Реализовать BDD подход

Цель: В результате выполнения дз вы поработаете и реализуете BDD подход , используя Page Object и Cucumber.

Реализовать 10 BDD-сценариев для сайта https://otus.ru/
Использовать Page Object и Cucumber
Уделить внимание удобству изменения Step Implementations и возможности переиспользовать шаги написанные на Gherkin

4 API Testing

1	**HTTP. Postman, Newman, Fiddler (Charles), curl, SOAP. SoapUI**	рассмотреть особенности протоколов HTTP и SOAP; использовать инструменты для ручного тестирования API на этих протоколах.
2	**RestAssured**	создать автоматизированные тесты API на Java.
3	**Использование API-helper'ов в UI-тестах**	объединить UI- и API-тесты для более эффективной автоматизации тестирования.

5 Параллельное выполнение тестов

1	**Параллельное выполнение тестов. Selenium Grid. Настройка и запуск**	рассмотреть подходы к распараллеливанию тестов; настроить и использовать базовые инструмент Selenium Grid.
2	**Современные способы распараллеливания. Selenoid**	рассмотреть современные подходы к распараллеливанию тестов; применить Selenoid. Домашние задания 1 Написать sh/bat скрипт, готовящий окрежение для параллельного тестирования Цель: В результате выполнения дз вы напишите и запустите скрипт, настроите все необходимые компоненты для параллельного выполнения тестов и сообщения номера прослушиваемого порта. Сценарий использования: запустить скрипт -> в консоль вывенден номер порта запустить тесты, используя RemoteWebDriver, указав в адресе порт из предыдущего шага
3	**Отчетность: Allure, Report Portal. Снятие скриншотов и запись видео**	организовать понятную отчетность по автотестам; внедрить инструменты для сбора логов тестов, скриншотов приложения и записи видео. Домашние задания 1 Добавить интеграцию с системой отчетности к своим тестам

6 Практики CI/CD

1	**Что такое CI/CD и зачем он нужен**	рассмотреть CI- и CD-процессы; рассмотреть основные инструменты и ключевые понятия (Server, agents, jobs. Fail fast, Scheduling, WebHooks).
2	**Jenkins**	настроить и запустить сервис. интегрировать с git и Docker; рассмотреть pipeline-подход.
3	**Описание шагов "от выгрузки до отчетов по тестированию". Scheduling, webhooks**	создать джобы для прогона автотестов от выгрузки из git до отображения отчета по тестированию; настроить запуск джобы по расписанию и по событию. Домашние задания 1 Доработать CI для удобной отчетности, выполнения по push в git и выполнять back-up конфигураций Цель: В результате выполнения дз вы запустите и настройте локально Jenkins (сервис или контейнер). Создайте job (можно использовать job, созданную в ходе выполнения прошлого домашнего задания) Шаги в job: 1. Выгрузить из вашего репозитория код тестов 2. Собрать проект 3. Выполнить все тесты 4. Прислать письмо вам на почту -- в письме указаны -- номер сборки -- статус сборки -- ветка реопзитория, из которой был взят код тестов -- колиечество тестов (всего/успешных/проваленных/пропущенных) -- общее время выполнения job'ы Настройте job так, чтобы она запускалась после каждого git push'a в ваш репозиторий (использовать webhooks) и каждую ночь в 01:00. Помимо отчетности по e-mail, отчет должен приходить в канал в slack Отчеты должны добавляться в систему отчетов (на ваш выбор allure, report portal и подобные) По окончанию выполнения job, должен выполняться back-up самой job'ы и настроек (можно использовать SCM Sync configuration plugin)
4	**Continuous Testing (Testing Pyramid, Test Metrics)**	

7 Проект A

1	**Stubs**	разработать "заглушки" на сторонние сервисы для большей изоляции автотестов.
2	**Обзор техонологий Appium, Selenide**	проанализировать основные инструменты и вспомогательные библиотеки, которые применяются сегодня в автоматизации; объяснить для чего они нужны и как работают.
3	**Курсовая работа**	реализовать проект автоматизации тестирования с применением имеющихся знаний и навыков для заданного приложения; защитить проект и получить рекомендации экспертов.

Домашние задания

1 Проектная работа от EPAM

Цель: Задача для проекта: Необходимо построить фреймворк для автоматизации E2E тестирования сайта с обязательным тестовым покрытием. Обязательным является использование библиотеки Healenium и Report Portal. Задачи с пометкой «*дополнительно» выполняются по желанию.

Что будем тестировать: Приложение https://events.epam.com/ предоставляет информацию о мероприятиях, которые проводит EPAM. Сайт позволяет посмотреть предстоящие/прошедшие мероприятия в разных городах, информацию о спикерах, докладах, календарь мероприятий.
Требования к фреймворку:
1. Java + Maven/Gradle + TestNG/Junit 5 проект
2. Для стабилизации нахождения локаторов используется библиотека Healenium (https://github.com/healenium/healenium-web)
3. Подключен Report Portal (https://reportportal.io/)
4. Настроено логирование
5. Реализована возможность кроссбаузерного тестирования и удаленного запуска тестов
6. Реализована возможность параллельного запуска тестов
7. Код проекта хранится в Git (важна частота и содержание коммитов)
8. Для работы со страницами используется паттерн Page Object
9. Код оформлен согласно Java Code Conventions, коментарии в стиле Javadoc приветствуются
*Дополнительно: Настроена интеграция с CI и запуск тестов по расписанию
Обязательное тестовое покрытие:
1. Просмотр предстоящих мероприятий:
1.1 Пользователь переходит на вкладку events
1.2 Пользователь нажимает на Upcoming Events
1.3 На странице отображаются карточки предстоящих мероприятий. Количество карточек равно счетчику на кнопке Upcoming Events

2. Просмотр карточек мероприятий:
2.1 Пользователь переходит на вкладку events
2.2 Пользователь нажимает на Upcoming Events
2.3 На странице отображаются карточки предстоящих мероприятий.
2.4 В карточке указана информация о мероприятии:
• город проведения, язык
• название мероприятия
• дата мероприятия
• информация о регистрации
• список спикеров
Важно проверить порядок отображаемых блоков с информацией в карточке мероприятия
3. Валидация дат предстоящих мероприятий:
3.1 Пользователь переходит на вкладку events
3.2 Пользователь нажимает на Upcoming Events
3.3 На странице отображаются карточки предстоящих мероприятий.
3.4 В блоке This week даты проведения мероприятий больше или равны текущей дате и находятся в пределах текущей недели.
4. Просмотр прошедших мероприятий в Канаде:
4.1 Пользователь переходит на вкладку events
4.2 Пользователь нажимает на Past Events
4.3 Пользователь нажимает на Location в блоке фильтров и выбирает Canada в выпадающем списке
4.4 На странице отображаются карточки прошедших мероприятий. Количество карточек равно счетчику на кнопке Past Events. Даты проведенных мероприятий меньше текущей даты.
5. Просмотр детальной информации о мероприятии:
5.1 Пользователь переходит на вкладку events
5.2 Пользователь нажимает на Upcoming Events
5.3 На странице отображаются карточки предстоящих мероприятий.
5.4 Пользователь нажимает на любую карточку
5.5 Происходит переход на страницу с подробной информацией о мероприятии
5.6 На странице с информацией о мероприятии отображается шапка с кнопкой для регистрации, программа мероприятия, карта
6. Фильтрация докладов по категориям:
6.1 Пользователь переходит на вкладку Talks Library
6.2 Пользователь нажимает на More Filters
6.3 Пользователь выбирает: Category – Testing, Location – Belarus, Language – English, На вкладке фильтров
6.4 На странице отображаются карточки соответствующие правилам выбранных фильтров
7. Поиск докладов по ключевому слову:
7.1 Пользователь переходит на вкладку Talks Library
7.2 Пользователь вводит ключевое слово QA в поле поиска
7.3 На странице отображаются доклады, содержащие в названии ключевое слово поиска
*Дополнительно: Тестовое покрытие может быть расширено для функциональности фильтрации