TokeroDevTestingPlaywright GitHub Repository Documentation

Introduction

TokeroDevTestingPlaywright repository contains a solution for automating web UI testing for tokero.dev website, utilizing Playwright technology with .NET C# to achieve this.

It includes multiple browser testing, multiple language testing and acceptable performance testing.

Installation

1. Clone the GitHub repository from: https://github.com/doritmtm/TokeroDevTestingPlaywright
2. Open Visual Studio 2022 with the solution TokeroDevTestingPlaywright.sln
3. Build the solution in Visual Studio
4. If using Playwright for the first time on the current computer, run `playwright.ps1 install` from binary folder (e.g. bin\Debug\net8.0). Playwright must first download the browsers it uses for testing before being able to run the tests.
5. Use Visual Studio Test Explorer to see and run the unit tests.

Policies list web page testing

The policies list web page testing is implemented in the tests folder in PoliciesListTest.cs:

1. Page_Title -> Tests the page tab title across all supported languages and browsers
2. Navigation_To_Page -> Tests if it is possible to navigate to policies list page from main home page across all supported languages and browsers
3. Page_List_Links -> Tests if the links to all policies are correct and displayed in the policies list page across all supported languages and browsers
4. Page_List_Names -> Tests if the names of the policies displayed in the policies list are correct for all supported languages and browsers
5. Page_List_Title -> Tests if the title of the list is correct for all supported languages and browsers. Currently, there is an issue on tokero.dev for German language and this test automatically detects it by failing.

Output

An example of successful output is shown here for Navigation_To_Page test:

Running Navigation_To_Page test on chromium for en language
Took 2901 ms
Running Navigation_To_Page test on chromium for ro language
Took 2373 ms
Running Navigation_To_Page test on chromium for de language
Took 2765 ms
Running Navigation_To_Page test on firefox for en language
Took 3682 ms
Running Navigation_To_Page test on firefox for ro language
Took 3316 ms
Running Navigation_To_Page test on firefox for de language
Took 3526 ms
Running Navigation_To_Page test on webkit for en language
Took 7562 ms
Running Navigation_To_Page test on webkit for ro language
Took 5932 ms
Running Navigation_To_Page test on webkit for de language
Took 4952 ms

An example of the fail of Page_List_Title test is shown here:

Test method TokeroDevTestingPlaywright.Tests.PoliciesListTests.Page_List_Title threw exception:
Microsoft.Playwright.PlaywrightException: Locator expected to contain text 'TOKERO Richtlinien und Regeln'
But was: 'TOKERO kasutustingimused ja reeglid'
Call log:
<...>

Stack Trace:
<...>

Standard Output:
TestContext Messages:
Running Page_List_Title test on chromium for en language
Took 3425 ms
Running Page_List_Title test on chromium for ro language
Took 4150 ms
Running Page_List_Title test on chromium for de language

Implementation documentation

StaticSettings.cs

The static settings contain the global settings for all tests and test running:

1. ShowBrowsers -> Whether Playwright should open and show browser windows or not. Playwright runs by default with the browsers hidden.
2. Width, Height -> The simulated window size of the browser
3. MainWebsiteUrl -> The home url of the website tested by all tests
4. Company -> Company name; Used to test the title of the site
5. MaxTimeToLoadPage -> The maximum number of milliseconds permitted for a page load. The page load time is multiplied in the tests to account for the initial page load and dynamic loading after page load
6. SupportedLanguages -> The list of supported languages with proper resource strings defined in the test project (Resources.resx).

TestHelper.cs

Helper functions for all tests designed to be reused in the actual tests:

1. InitializePage -> Initializes one browser with one page for the given browser type (Browser types can be Chromium, Firefox and Webkit, Playwright object properties)
2. CloseBrowserHostingPage -> Utility function to close and clean up open browser for the given page after finishing with it
3. RunAllLanguagesForBrowser -> Utility function that receives the test and runs it for all supported languages for the given browser type
4. RunOnAllBrowsersAllLanguages -> Main utility function to use for the testing that receives the test and runs it for all supported browsers and languages. The test must be written in a browser and language agnostic way.
5. GetLocalizedString -> gets the localized version of the string associated by the given id in Resources.resx for the given language
6. AcceptCookies -> Searches for the existence of the cookie dialog and clicks accept cookies if it could be found in 1000 ms. Should be used after waiting for the page to fully load.
7. WaitForFullPageLoad -> Makes Playwright wait for the full page load by waiting both for the page load event and an UI element in the header that appears after dynamic load (e.g. the language switcher)

Resources.cs

An empty class used to publicly attach and use and point to the proper Resources.resx files.

In the Resources.resx file it can be added strings that are localized to multiple languages. Here it can be defined what text the tests should expect for each language

for the id defined under Name column. Use these ids to make the tests language agnostic.

To add more languages, simply create more resource files. (e.g. Resources.fr.resx)

MSTestSettings.cs

Here it is defined whether the tests should be run in parallel or not. Currently it is configured to run serially to avoid timeouts and out of memory problems. Parallelism is also stopped on each TestClass by specifying alongside it the attribute DoNotParallelize.

Tests folder

Here specific tests shall be created and defined. It is recommended to create a cs file for each tested web page.

Troubleshooting

There were some race conditions and syncing to be resolved in the tests:

1. Waiting for the full page load
2. Cookies dialog appears after a time after full page load
3. The element to be clicked must be in view
4. We cannot directly test the page link immediately after clicking on an element that triggers a navigation

While Playwright should be able to handle these things automatically, Playwright wasn't designed for optional elements. Therefore, we must manually check for the cookie dialog as it will later interfere with clicking elements and with finding cookie elements.

Some race condition issue was present only for WebKit browser testing where sometimes the click on an element not in view would not perform the click. Therefore, we manually instruct Playwright to scroll it into view and later click it. Playwright was scrolling by itself on click function, but not stable unfortunately.

Regarding navigation link testing issue, this happened only for WebKit browser testing as Playwright should handle waiting for navigation itself. We now specifically instruct Playwright to wait for the navigation.