WebUiAutomationToolKit

Monday, February 4, 2019 8:50 AM

WebUiAutomationToolKit.dll

The assembly that contains all class and interface definitions for this utility.

WebUiAutomationToolKit.xml

Xml documentation file that contains all the summary method documentation for Visual Studio intellisense

WebUiAutomation

Contains the utility and component instance creation methods for automating cross browser testing of a web site. This component is the only one that should be directly accessed and be 'newed up'.

Public constructor

- WebUiAutomation(string driverPath = null, int timeoutForWaitOverride = null)
  - driverPath -> This is the path where that driver service executables for each browser can be accessed from. This path is verified and, if it exists, will be checked for the appropriate executables.
    - If the driverPath parameter is null, the framework will look in the current directory.
  - timeoutForWaitOverride -> The default for this is 5 seconds and it is and optional parameter. It can be overridden by providing a positive, non-zero integer.

Methods

- IWebDriverManager GetIWebDriverManager()
  - Creates and returns and instance of the IWebDriverManager used to create and manage IWebDriver objects

- ILogger GetLogger(LoggerSettings loggerSettings)
  - Creates and returns and instance of the ILogger used to write to a common log file
  - loggerSettings -> Settings for the logger

- ITestExecutor GetTestExecutor(string resultsPath, bool collectTestData = false)
  - Creates and returns and instance of the ITestExecutor used to execute tests and manage the results
  - collectTestData -> Passing true will enbale the built in test data collection module
  - resultsPath -> This is where screenshots taken during failures will be saved
  - Calling ITestExecutor.GetTestSummary() at the end of a test run will return a detailed summary of the current run

- string ParseInnerText(string htmlText)
  - This method will remove html data from the innerText of a given element and return only text NOT go beyon the first element. This method will not parse the innerText of a child element passed in as part of a larger set
  - htmlText -> Raw htmlText from an IWebElement that contains html to be removed

- ReadOnlyCollection<IWebElement> GetAllElementsUsingMatchingSelectorData(SelectorData selectorData, IWebDriverManager webDriverManager)
  - Finds and returns all elements matching the provided selector data
  - selectorData -> data to check for in the current DOM instance

ILogger

- Thread safe logger to log test related messages according to a defined set of message types

Methods

- void Log(LogMessageType level, string message)
  - level -> One the values defined by in WebAutomationEnums.LogMessageType
  - Message -> The message to be logged

    Date information is added by the logger automatically. For example a message of 'Clicking Submit' with MessageType TESTINFO, logged within a test method named 'Execute' would be written as:

    1/25/2019 10:52:09 AM - Message Type:TESTINFO Test Method:Execute - Clicking Submit

- string ToXML(object obj);
  - Serializes the provided object to xml then returns the string representation of that xml serialization

- string GetCurrentLogFileName
  - Returns the name of the failure log file currently in use

- string GetCurrentFailureLogFileName
  - Returns the name of the failure log file currently in use

ITestExecutor

- This interface provides access to thread safe test execution delegate methods.
  - These methods have built in logging to output test start, pass and failure messaging as well as exception logging.
  - Any exception thrown within the execute method will be logged and then rethrown to be handled by the caller.

Methods

- void Execute(Action testMethod, IWebDriverManager webDriverManager)
  - testMothod -> This is the test method to execute
  - webDriverManager -> This should have at least one driver instance created otherwise execution will fail and an exception will be thrown
- string GetTestRunSummary()
  - Returns formatted string of test results that can be written to a log file or elsewhere

IWebDriverManager

- This interface provides access to the methods available on the IWebDriver interface as well as the ability manage multiple driver instances

Methods

- void SetDriverOptions(ChromeOptions chromeOptions = null, FirefoxOptions firefoxOptions = null, InternetExplorerOptions internetExplorerOptions = null, EdgeOptions edgeOptions = null)
  - Sets the options for the associated driver. If the driver is active when this method is called it will be recreated
  - If no options objects are set the default options defined on the driver will be used

- void SetFirefoxProfile(FirefoxProfile firefoxProfile)
  - Sets the profile for the firefox driver. If the driver is active when this method is called it will be recreated.
  - If no profile object is set the default profile defined on the browser will be used

- void CreateDriverInstance(DriverType driverType)
  - Creates an instance of IWebDriver that matches the type provided

- void QuitDriverInstance(DriverType driverType)
  - Quits an instance of IWebDriver that matches the type provided

- void SetActiveDriver(DriverType driverType)
  - Selects the driver to mark as active. The active driver is used for execution.

- DriverType GetActiveDriverType()
  - Returns the DriverType value indicating what driver is currently marked as active. The active driver is used for execution.

- string GetActiveDriverPageSource()
  - Gets the source of the page last loaded by the browser.

- string GetActiveDriverTitle()
  - Gets the title of the current browser window

- string GetActiveDriverUrl()
  - Gets or sets the URL the browser is currently displaying.

- string GetActiveDriverCurrentWindowHandle()
  - Gets the current window handle, which is an opaque handle to this window that uniquely identifies it within this driver insta nce.

- ReadOnlyCollection<string> GetActiveDriverWindowHandles()
  - Gets the window handles of open browser windows.

- void NavigateWithActiveDriver(string url)
  - Instructs the driver to navigate the browser to another location.

- void CloseLastTabWithActiveDriver(string mainWindow)
  - Closes the farthest tab to the right in the current browser window

- void CloseTabWithActiveDriver(string tabToClose, string targetTab)
  - Closes the tab designated in the tabToClose parameter and switches the context to the tab designated in the targetTab paramet er

- void SwitchToLastTabWithActiveDriver()
  - Switches the context to the farthest tab to the right in the current browser window

- void CloseAllTabsExceptCurrentWithActiveDriver()
  - Closes all tabs except for the one currently with focus

- object ExecuteScript(string script, params object[] args)
  - Executes JavaScript in the context of the currently selected frame or window

- object ExecuteAsyncScript(string script, params object[] args)
  - Executes JavaScript asynchronously in the context of the currently selected frame or window

- IWebElement CheckChildElementExistsAndReturnIt(SelectorData parentSelectorData, SelectorData childSelectorData, IWebDriverManager webDriverManager, int nthParentElement = -1)
  - Finds and returns the child IWebElement using the parameters provided, if none is found null is returned
  - parentSelectorData -> Data to find the parent element with
  - childSelectorData -> Data to find the child element with
  - nthParentElement > The Zero based position of the parent element to search with, this is optional

- By CheckElementExistsReturnCssSelector(SelectorData selectorData, IWebDriverManager webDriverManager)
  - Finds the IWebElement using the parameters provided and returns the CssSelector based By object, if none is found null is ret urned
  - selectorData -> Data to build the CssSelector with

- IWebElement CheckElementExistsReturnIWebElement(SelectorData selectorData, IWebDriverManager webDriverManager)
  - Finds and returns the IWebElement using the parameters provided, if none is found null is returned
  - selectorData -> Data to build the CssSelector with

- bool CheckElementsExist(SelectorDataSet selectorDataSet, IWebDriverManager webDriverManager)
  - Builds a CssSelector with the SelectorDataSet provided and uses it to check that an element exists with that data
  - selectorDataSet -> Data to check for in the current DOM instance

- IWebDriverManager Clear(IWebDriverManager webDriverManager, SelectorData selectorData)
  - Clears the text from the provided element after verifiying the element is visible
  - selectorData -> Object representing the element to be cleared

- IWebDriverManager Click(IWebDriverManager webDriverManager, SelectorData selectorData)
  - Clicks the provided element after verifying it exists and is clickable
  - This method will retry the click once after waiting 2 seconds after the first ElementClickInterceptedException is thrown
  - selectorData -> Object representing the element to click

- bool bool DoesUrlContainUsingRegex(IWebDriverManager webDriverManager, string pattern)
  - Checks whether or not the current url contains the provided pattern using regex IsMatch

- bool DoesUrlContain(IWebDriverManager webDriverManager, string text)
  - Checks whether or not the current url contains the provided text
  - text -> Text to look for in the current Url

- void HighlightElement(IWebDriverManager webDriverManager, SelectorData selectorData)
  - Modifies the style associated with the selector data object provided to highlight the element on the page
  - selectorData -> Object representing the element to highlight

- bool IsElementVisible(IWebDriverManager webDriverManager, SelectorData selectorData)
  - Checks whether or not the provided element is visible(This requires IWebElement.Displayed and IWebElement.Enabled to both eva luate to true)
  - selectorData -> Object representing the element to check

- void JavaScriptClick(IWebDriverManager webDriverManager, SelectorData selectorData)

        - Clicks the element associated with the selector data object provided using a JavaScript query. This is useful when the elemen t being clicked may be obstructed
        - selectorData -> Object representing the element to highlight

    - IWebDriverManager MoveToElement(IWebDriverManager webDriverManager, SelectorData selectorData)
        - Clears the text from the provided element after verifiying the element is visible
        - selectorData -> Object representing the element to move to

    - string GetCurrentLogFileName
        - Returns the name of the failure log file currently in use

    - string GetCurrentLogFileName
        - Returns the name of the log file currently in use

    - IWebDriverManager SendText(IWebDriverManager webDriverManager, SelectorData selectorData, string textToEnter)
        - Enters text into the provided element after verifying it exists and is visible
        - selectorData -> Object representing the element to receive the text
        - textToEnter -> Text that will be entered

    - IWebDriverManager SendTextWithDelay(IWebDriverManager webDriverManager, SelectorData selectorData, string textToEnter, int delay = 500)
        - Enters text into the provided element one character at a time with a delay between each after verifying it exists and is visi ble. Note: Only use this method to send text to elements that have autocomplete and require a delay on each letter typed.
        - selectorData -> Object representing the element to receive the text
        - textToEnter -> Text that will be entered
        - delay -> Interval before each character is entered

    - void TakeScreenShot(IWebDriverManager webDriverManager, string sreenShotPath, string screenShotName)
        - Takes a screenshot and saves it in the provided path. The file name is in the form of "{testMethodName}_{DateTime.Now}.jpeg"
        - sreenShotPath -> Path to save the screenshot to
        - screenShotName -> Name of the screenshot file

    - List<KeyValuePair<By, WebAutomationEnums.NavigationResult>> TestLinkNavigationForAllAnchorsFoundInPage(IWebDriverManager webDriverManager)
        - Validates that all the anchor tags <a /> found on the page the driver is currently navigated to. It returns the XPath By obje ct and NavigationResult for every link found and tested

    - bool IsElementDisplayed(IWebDriverManager webDriverManager, SelectorData selectorData)
        - Checks whether or not the provided element is visible(This requires IWebElement.Displayed evaluate to true)
        - selectorData -> Object representing the element to check

    - bool IsElementEnabled(IWebDriverManager webDriverManager, SelectorData selectorData)
        - Checks whether or not the provided element is enabled(This requires IWebElement.Enabled evaluate to true)
        - selectorData -> Object representing the element to check

    - bool IsElementDisplayedAndEnabled(IWebDriverManager webDriverManager, SelectorData selectorData)
        - Checks whether or not the provided element is visible(This requires IWebElement.Displayed and IWebElement.Enabled to both eva luate to true)
        - selectorData -> Object representing the element to check

    - void TakeElementScreenShot(string sreenShotPath, string screenShotName, SelectorData selectorData)
        - Takes a screenshot of an element in the current browsing context and saves it in the provided path.
        - The file name is in the form of "{testMethodName}_{DateTime.Now}.jpeg"
        - sreenShotPath ->Path to save the screenshot to
        - screenShotName ->Name of the screenshot file
        - selectorData :->Element selector data to find screenshot target

IDomTree

- This interface provides access to the methods available for interacting with objects of type IDomTree

Methods

- void Build(SelectorData rootNode, IWebDriverManager webDriverManager)
    - Builds a tree structure beginning with the SelectorData provided as the root node

- IDomNode MoveToFirstChild(IWebDriverManager webDriverManager)
  - Moves to the first(when reading the DOM top to Bottom) child node of the current node. If no child is found, null is returned

- IDomNode MoveToNthChild(IWebDriverManager webDriverManager, int nthChild)
  - Moves to the nth(when reading the DOM top to Bottom) child node of the current node. If no child is found, null is returned

- IDomNode MoveToParent(IWebDriverManager webDriverManager)
  - Moves to the parent node of the current node. If no parent is found or if the current node is the root node, null is returned

- IDomNode NextSibling(IWebDriverManager webDriverManager)
  - Moves to the next(when reading the DOM top to Bottom) sibling node of the current node. If no sibling is found or if the curr ent node is the root node, null is returned

- IDomNode PreviouSibling(IWebDriverManager webDriverManager)
  - Moves to the previous(when reading the DOM top to Bottom) sibling node of the current node. If no sibling is found or if the current node is the root node, null is returned

- void SwitchToFrame(SelectorData selectorData)
  - Switches the driver context to the frame matching the SelectorData provided

IDomNode

- This interface provides access to the methods available for interacting with objects of type IDomNode

Methods

- Dictionary<string, object> GetAttributes(IWebDriverManager webDriverManager)
  - Returns any attributes found in the form of a dictionary where the keys are attributes names and the values are the attribute values

- string GetTagName(IWebDriverManager webDriverManager)
  - Gets the tag name of this node

- IDomNode MoveToNthChild(IWebDriverManager webDriverManager, int nthChild)
  - Moves to the nth(when reading the DOM top to Bottom) child node of the current node. If no child is found, null is returned

- string GetText(IWebDriverManager webDriverManager)
  - Gets the innerText of this node, without any leading or trailing whitespace, and with other whitespace collapsed

- bool HasAttributes(IWebDriverManager webDriverManager)
  - Indicates if this node has attributes or not

- bool HasChildren(IWebDriverManager webDriverManager)
  - Indicates if this node has child nodes or not