```
// ***********************************************************************
// Assembly         : GeneralFunctionality
// Author           : G.H.M.H. Schmeits
// Created          : 01-05-2018
//
// Last Modified By : G.H.M.H. Schmeits
// Last Modified On : 05-15-2019
// ***********************************************************************
// <copyright file="Functions.cs" company="SCHMEITS SOFTWARE">
//     Copyright ©  2018
// </copyright>
// <summary></summary>
// ***********************************************************************

using DataStorage;
using Microsoft.Win32;
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Interactions;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.Extensions;
using OpenQA.Selenium.Support.UI;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Management;
using System.Text;
using System.Threading;
using System.Windows.Forms;
using System.Xml;
using System.Xml.Linq;
using Keys = OpenQA.Selenium.Keys;

namespace GeneralFunctionality
{
    /// <summary>
    ///     Class Functions.
    /// </summary>
    /// <autogeneratedoc />
    /// TODO Edit XML Comment Template for Functions
    public class Functions
    {
        /// <summary>
        ///     The credentials
        /// </summary>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for credentials
        public static Credentials credentials;

        /// <summary>
        ///     The database connection
        /// </summary>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for databaseConnection
        public static DatabaseConnection databaseConnection;
        /// <summary>
        ///     The serial key XML
        /// </summary>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for serialKeyXml
        public static SerialKeyXML serialKeyXml;

        /// <summary>
        ///     The testrun
        /// </summary>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for _testrun
        private static string _testrun;


        public static string _project;

        /// <summary>
        ///     The application name
        /// </summary>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for applicationName
        private static string applicationName;

        /// <summary>
        ///     The class name
        /// </summary>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for className
        private static string className;

        /// <summary>
        ///     The counter check
        /// </summary>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for counterCheck
        private static int counterCheck;

        /// <summary>
        ///     The testnr
        /// </summary>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for testnr
        private static string testnr;

        public static bool uitgevoerd { get; set; }

        /// <summary>
        ///     Gets or sets the HTML page.
        /// </summary>
        /// <value>The HTML page.</value>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for HtmlPage
        public static string HtmlPage { get; set; }

        /// <summary>
        ///     Gets or sets the name of the screenshot.
        /// </summary>
        /// <value>The name of the screenshot.</value>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for ScreenshotName
        public static string ScreenshotName { get; set; }

        /// <summary>
        ///     Gets or sets the set testnr.
        /// </summary>
        /// <value>The set testnr.</value>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for SetTestnr
        private static string SetTestnr { get; set; }

        /// <summary>
        ///     Checks the by.
        /// </summary>
        /// <param name="driver">The driver.</param>
        /// <param name="idString">The identifier string.</param>
        /// <param name="checkedBy">The checked by.</param>
        /// <param name="testnr">The testnr.</param>
        /// <param name="browserid">The browserid.</param>
        /// <param name="tekst">The tekst.</param>
        /// <param name="action">The action.</param>
        /// <param name="url">The URL.</param>
        /// <param name="switch1">The switch1.</param>
        /// <param name="description">The description.</param>
        /// <param name="inverseString">The inverse string.</param>
        /// <param name="comment">The comment.</param>
        /// <param name="testname">The testname.</param>
        /// <param name="testcase">The testcase.</param>
        /// <param name="version">The version.</param>
        /// <param name="counter">The counter.</param>
        /// <param name="machinenumber">The machinenumber.</param>
        /// <param name="testelementname">The testelementname.</param>
        /// <param name="start_url">The start URL.</param>
        /// <param name="texttext">The texttext.</param>
        /// <param name="testpage">The testpage.</param>
        /// <param name="testscenario_name">Name of the testscenario.</param>
        /// <returns><c>true</c> if XXXX, <c>false</c> otherwise.</returns>
        public static bool CheckBy(
            IWebDriver driver,
            string idString,
            string checkedBy,
            string testnr = "1",
            string browserid = "1",
            string tekst = "",
            string action = "",
            string url = "",
            string switch1 = "",
            string description = "",
            string inverseString = "",
            string comment = "",
            string testname = "",
            string testcase = "",
            string version = "",
            int counter = 0,
            string machinenumber = "",
            string testelementname = "",
            string start_url = "",
            string texttext = "",
            string testpage = "",
            string testscenario_name = "",
            bool opslaan = true,
            bool strikt = true)
        {
            bool present;
            var opslaan1 = opslaan;

            present = true;

            if (checkedBy != string.Empty || idString != string.Empty)
            {
                var showBorderWidth = OverallSettings.ShowBorderWidth;
                var showShowTime = OverallSettings.ShowDuration;
                var showBorderColor = OverallSettings.ShowBorderColor;
                var extraTimes = OverallSettings.ShowExtraChecks;

                SetTestnr = testnr;
                ScreenshotName = string.Empty;
                var gewoon = 3;
                var ongewoon = 4;
                if (inverseString == "yes")
                {
                    gewoon = 4;
                    ongewoon = 3;
                }

                var result = ongewoon;

                try
                {
                    IWebElement content = null;
                    var actions = new Actions(driver);

                    switch (checkedBy.ToUpper())
                    {
                        case "NAME":
                            driver.FindElement(By.Name(idString));
                            content = driver.FindElement(By.Name(idString));
                            break;
                        case "XPATH":
                            driver.FindElement(By.XPath(idString));
                            content = driver.FindElement(By.XPath(idString));
                            break;
                        case "CLASS":
                            driver.FindElement(By.ClassName(idString));
                            content =
                                driver.FindElement(By.ClassName(idString));
                            break;
```

```
    case "ID":
        driver.FindElement(By.Id(idString));
        content = driver.FindElement(By.Id(idString));
        break;
    case "CSSSELECTOR":
        driver.FindElement(By.CssSelector(idString));
        content =
            driver.FindElement(By.CssSelector(idString));
        break;
}

if (content != null && tekst != string.Empty &&
    content.Text != string.Empty)
{
    if (strikt)
    {
        if (tekst != string.Empty &&
            content.Text == tekst)
            result = gewoon;
    }
    else
    {
        if (tekst != string.Empty &&
            content.Text.IndexOf(tekst) != -1)
            result = gewoon;
    }

    /*if (tekst != string.Empty &&
        content.Text != tekst)
        result = ongewoon;*/
}

/*
if (content != null && tekst != string.Empty &&
    content.TagName == "input" &&
    content.GetAttribute("value") != string.Empty)

    //if (content != null && tekst != string.Empty &&
    //(cont/ent.TagName == "input" || content.TagName == "textarea") &&
    //content.GetAttribute("value") != string.Empty)
{
    if (strikt == true)
    {
        if (tekst != string.Empty &&
            content.GetAttribute("value").Trim() == tekst)
            result = gewoon;
    }
    else
    {
        if (tekst != string.Empty &&
            content.GetAttribute("value").Trim().IndexOf(tekst) != -1)
        result = gewoon;
    }
```

```

                        if (tekst != string.Empty &&
                            content.GetAttribute("value").Trim() != tekst)
                            result = ongewoon;
                    }
                    */


                    /*if (content != null && content.Text != null &&
                        content.Text != string.Empty &&
                        tekst != string.Empty)
                    {
                        // MessageBox.Show("|" + content.Text.Trim() + "|" + tekst.Trim() + "|");

                        if (tekst != string.Empty &&
                            content.Text.Trim() == tekst.Trim())
                            result = gewoon;

                        if (tekst != string.Empty &&
                            content.Text.Trim() != tekst.Trim())
                            result = ongewoon;
                    }*/

                    if (tekst == string.Empty) result = gewoon;

                    HighlightAndScreenshot(driver, content, showShowTime,
                        showBorderColor, showBorderWidth);

                    if (idString.ToUpper().Trim() != "POPUP" &&
                        idString.Trim().Length > 0)
                        if (result == ongewoon)
                            TakeScreenshot(driver, ongewoon);

                    present = true;
                }
                catch (NoSuchElementException e)
                {
                    if (gewoon == 3)
                    {
                        present = false;
                        TakeScreenshot(driver, result);


                        if (counter < extraTimes)
                        {
                            counter++;
                            CheckBy(
                                driver,
                                idString,
                                checkedBy,
                                testnr,
                                browserid,
                                tekst,
                                action,
                                url,
                                switch1,
                                description,
                                version,
                                inverseString,
                                comment,
                                testname,
                                testcase,
                                counter,
                                machinenumber,
                                testelementname,
                                start_url,
                                texttext,
                                testpage,
                                testscenario_name,
                                opslaan1,
                                strikt);
                        }
                    }
                    else
                    {
                        present = true;
                    }
                }
                finally
                {
                    applicationName = testname;
                    if (result == gewoon) ScreenshotName = string.Empty;
                    if (idString.ToUpper().Trim() == "POPUP") result = gewoon;

                    if (opslaan)
                        TestResults.AddTestResult(
                            applicationName,
                            className,
                            _testrun,
                            Functions.testnr,
                            idString,
                            checkedBy,
                            result.ToString(),
                            ScreenshotName,
                            action,
                            tekst,
                            url,
                            switch1,
                            description,
                            browserid,
                            inverseString,
                            comment,
                            version,
                            machinenumber,
                            testelementname,
                            start_url,
                            texttext,
                            "",
                            testpage,
                            Convert.ToString(getProjectID())
                        );
                }
            }

            return present;
        }

        /// <summary>
        ///     Gets the HDD serial number.
        /// </summary>
        /// <returns>System.String.</returns>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for getHddSerialNumber
        public static string getHddSerialNumber()
        {
            // --- Win32 Disk
            var searcher = new ManagementObjectSearcher("\\root\\cimv2",
                "select * from Win32_DiskPartition WHERE BootPartition=True");

            uint diskIndex = 999;
            foreach (ManagementObject partition in searcher.Get())
            {
                diskIndex =
                    Convert.ToUInt32(partition.GetPropertyValue("Index"));
                break; // TODO: might not be correct. Was : Exit For
            }

            // I haven't found the bootable partition. Fail.
            if (diskIndex == 999)
                return string.Empty;

            // --- Win32 Disk Drive
            searcher = new ManagementObjectSearcher(
                "SELECT * FROM Win32_DiskDrive where Index = " + diskIndex);

            var deviceName = "";
            foreach (ManagementObject wmi_HD in searcher.Get())
            {
                deviceName = wmi_HD.GetPropertyValue("Name").ToString();
                break; // TODO: might not be correct. Was : Exit For
            }

            // I haven't found the disk drive. Fail
            if (string.IsNullOrEmpty(deviceName.Trim()))
                return string.Empty;

            // -- Some problems in query parsing with backslash. Using like operator
            if (deviceName.StartsWith("\\\\.\\"))
                deviceName = deviceName.Replace("\\\\.\\", "%");

            // --- Physical Media
            searcher = new ManagementObjectSearcher(
                "SELECT * FROM Win32_PhysicalMedia WHERE Tag like '" +
                deviceName +
                "'");
            var serial = string.Empty;
            foreach (ManagementObject wmi_HD in searcher.Get())
            {
                serial = wmi_HD.GetPropertyValue("SerialNumber").ToString();
                break; // TODO: might not be correct. Was : Exit For
            }

            return serial;
        }


        public static int getMachineCode()
        {
            var searcher =
                new ManagementObjectSearcher("select * from
                  Win32_Processor");
            var collectedInfo = "";
            // here we will put the informa
            foreach (ManagementObject share in searcher.Get())
                // first of all, the processorid
                collectedInfo += share.GetPropertyValue("ProcessorId");

            searcher.Query = new ObjectQuery("select * from Win32_BIOS");
            foreach (ManagementObject share in searcher.Get())
                //then, the serial number of BIOS
                collectedInfo += share.GetPropertyValue("SerialNumber");

            searcher.Query = new ObjectQuery("select * from
              Win32_BaseBoard");
            foreach (ManagementObject share in searcher.Get())
                //finally, the serial number of motherboard
                collectedInfo += share.GetPropertyValue("SerialNumber");

            // patch luca bernardini
            if (string.IsNullOrEmpty(collectedInfo) | (collectedInfo == "00")
               |
                (collectedInfo.Length <= 3))
                collectedInfo += getHddSerialNumber();

            return getEightByteHash(collectedInfo, 100000);
        }

        /// <summary>
        ///     Gets the eight byte hash.
        /// </summary>
        /// <param name="s">The s.</param>
        /// <param name="MUST_BE_LESS_THAN">The must be less than.</param>
        /// <returns>System.Int32.</returns>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for getEightByteHash
        public static int getEightByteHash(string s,
            int MUST_BE_LESS_THAN = 1000000000)
        {
            //This function generates a eight byte hash

            //The length of the result might be changed to any length
            //just set the amount of zeroes in MUST_BE_LESS_THAN
            //to any length you want
            uint hash = 0;

            foreach (var b in Encoding.Unicode.GetBytes(s))
            {
                hash += b;
                hash += hash << 10;
                hash ^= hash >> 6;
            }

            hash += hash << 3;
            hash ^= hash >> 11;
            hash += hash << 15;

            var result = (int)(hash % MUST_BE_LESS_THAN);
            var check = MUST_BE_LESS_THAN / result;

            if (check > 1) result *= check;

            return result;
        }


        /// <summary>
        ///     Checks the screenshot dir.
        /// </summary>
        /// <param name="dirName">Name of the dir.</param>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for CheckScreenshotDir
        public static void CheckScreenshotDir(string dirName = "screenshots")
        {
            var screenshotsPath = GetCurrentDir(1);
            var pathG = screenshotsPath.Split('\\');
            var g = string.Empty;

            for (var i = 0; i < pathG.Length - 2; i++) g += pathG[i] + "\\";

            screenshotsPath = g;
            var di = new DirectoryInfo(screenshotsPath);
            var dis = new DirectoryInfo(screenshotsPath + "\\" + dirName);
            var diserror = new DirectoryInfo(screenshotsPath + "\\" + dirName);
            if (!dis.Exists)
            {
                var driverPathSub = di.CreateSubdirectory(dirName);
                var applicationPathSub =
                    di.CreateSubdirectory(dirName + "\\" +
                                          applicationName.Trim());
                var applicationPathSubErrors =
                    di.CreateSubdirectory(dirName + "\\" +
                                          applicationName.Trim());
            }
            else
            {
                var applicationPath =
                    new DirectoryInfo(screenshotsPath + "\\" + dirName + "\\" +
                                      applicationName.Trim());
                if (applicationPath.Exists)
                {
                }
                else
                {
                    var applicationPathSub =
                        di.CreateSubdirectory(dirName + "\\" +
                                              applicationName.Trim());
                }

                applicationPath = new DirectoryInfo(screenshotsPath + "\\" +
                                                    dirName + "\\" +
                                                    applicationName.Trim());

                if (!applicationPath.Exists)
                {
                    var applicationPathSubErrors =
                        di.CreateSubdirectory(dirName + "\\" +
                                              applicationName.Trim());
                }
            }
        }

        public static void CheckDir(string project)
        {
            var projectPath = GetCurrentDir(1) + project + @"\CSV";

            var pathG = projectPath.Split('\\');
            var g = string.Empty;

            for (var i = 0; i < pathG.Length - 1; i++) g += pathG[i] + "\\";

            projectPath = g;
            var di = new DirectoryInfo(projectPath);
            var dis = new DirectoryInfo(projectPath + "\\CSV");
            var diserror = new DirectoryInfo(projectPath + "\\CSV");
            if (!dis.Exists)
            {
                var driverPathSub = di.CreateSubdirectory("CSV");
                var applicationPathSub = di.CreateSubdirectory("CSV\\");
                var applicationPathSubErrors = di.CreateSubdirectory("CSV\\");
            }
            else
            {
                var applicationPath =
                    new DirectoryInfo(projectPath + "\\CSV");
                if (applicationPath.Exists)
                {
                }
                else
                {
                    var applicationPathSub = di.CreateSubdirectory("\\CSV");
                }

                applicationPath = new DirectoryInfo(projectPath + "\\CSV");

                if (!applicationPath.Exists)
                {
                    var applicationPathSubErrors =
                        di.CreateSubdirectory("CSV\\");
                }
            }
        }


        /// <summary>
        ///     Creates the HTML.
        /// </summary>
        /// <param name="testrun">The testrun.</param>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for CreateHTML
        public static void CreateHTML(string testrun)
        {
            try
            {
                var totalTestResultsHtmls =
                    new List<TestResults.TotalTestResultsHTML>();
                totalTestResultsHtmls =
                    TestResults.GetTotalTestResultsHTML(testrun);

                var testResults = new List<TestResults.TestResultsHTML>();
                testResults = TestResults.GetTestResultsHTML(testrun);
                setApplicationName(testResults[0].application);

                var timestring = "run_" +
                                 string.Format("{0,8:D8}",
                                     Convert.ToInt32(testrun));
                var message = "<!DOCTYPE html>\r\n<html>\r\n<head>";
                message += "\t<meta charset=\"UTF-8\">\r\n";
                message +=
                    "\t<meta name=\"viewport\" content=\"width=device-width,initial-scale=1\">\r\n";
                message += "\t<title>Testrun: " + testResults[0].application +
                           "_" + timestring + "</title>\r\n";
                message += "\t<link rel=\"stylesheet\" href=\"" +
                           GetCurrentDir(0) + "\\style.css\">\r\n";
                message += "</head>";
                message +=
                    "<body class=\"dark-theme\"><script id=\"__bs_script__\">//<![CDATA[document.write(\"<script async src='/browser-sync/browser-sync-client.js?v=2.23.6'><\\/script>\".replace(\"HOST\", location.hostname));//]]></script>\r\n";
                message += "<div id=\"wrapper\">\r\n";
                message += "\t<h1>Test: " + testResults[0].application.Trim() +
                           "</h1>\r\n";
                message += "\t<h2>Testrun: " +
                           testResults[0].testrun.ToString().Trim() +
                           "</h2>\r\n";
                message += "\t<h2>Browser: " +
                           totalTestResultsHtmls[0].testrun_browser +
                           "</h2>\r\n";
                message += "\t<div id=\"dashboard\">\r\n";

                // TestResults total
                var passedInt =
                    Convert.ToDouble(totalTestResultsHtmls[0].testrun_passed);
                var failedInt =
                    Convert.ToDouble(totalTestResultsHtmls[0].testrun_failed);
                var passedAfgerond =
                    Convert.ToInt32(passedInt / (passedInt + failedInt) * 100 -
                                    0.50);
                var passedAfgerond1 = string.Format("{0:F1}",
                    passedInt / (passedInt + failedInt) * 100);
                message += "\t\t<div class=\"row\">\r\n";

                // Grafic
                message += "\t\t\t<div class=\"col-4\">\r\n";
                message += "\t\t\t\t<div class=\"card\">\r\n";
                message +=
                    "\t\t\t\t\t<div class=\"card__content row donut\">\r\n";
                message +=
                    "\t\t\t\t\t\t<div id=\"chart\" class=\"col-5\" data-percent=\"" +
                    passedAfgerond1
                    + "\">\r\n";
                message +=
                    "\t\t\t\t\t\t\t<svg width=\"100%\" height=\"100%\" viewbox=\"0 0 40 25\" class=\"donut\">\r\n";
                message +=
                    "\t\t\t\t\t\t\t\t<circle class=\"donut-ring\" cx=\"20\" cy=\"13\" r=\"16\" fill=\"transparant\"></circle>\r\n";
                message += "\t\t\t\t\t\t\t\t<circle class=\"donut-segment p-" +
                           passedAfgerond
                           + "\" cx=\"20\" cy=\"13\" r=\"16\" fill=\"transparant\" stroke-dashoffset=\"25\"></circle>\r\n";
                message += "\t\t\t\t\t\t\t</svg>\r\n";
                message += "\t\t\t\t\t\t</div>\r\n";
                message += "\t\t\t\t\t\t<div class=\"col-7\">\r\n";
                message += "\t\t\t\t\t\t\t<ul id=\"runtime\">\r\n";
                message +=
                    "\t\t\t\t\t\t\t\t<li><strong>Start time: </strong></br>" +
                    string.Format(
                        "{0:dd/MM/yy HH:mm:ss}",
                        totalTestResultsHtmls[0].testrun_begintime) +
                    "</li>\r\n";
                message +=
                    "\t\t\t\t\t\t\t\t<li><strong>End Time: </strong></br>" +
                    string.Format(
                        "{0:dd/MM/yy HH:mm:ss}",
                        totalTestResultsHtmls[0].testrun_endtime) + "</li>\r\n";
                message +=
                    "\t\t\t\t\t\t\t\t<li><strong>Execution Time: </strong></br>"
                    + string.Format("{0:HH:mm:ss}",
                        totalTestResultsHtmls[0].testrun_time) + "</li>\r\n";
                message += "\t\t\t\t\t\t\t</ul>\r\n";
                message += "\t\t\t\t\t\t</div>\r\n";
                message += "\t\t\t\t\t</div>\r\n";
                message += "\t\t\t\t</div>\r\n";
                message += "\t\t\t</div>\r\n";

                // Passed
                message += "\t\t\t<div class=\"col-4\">\r\n";
                message += "\t\t\t\t<div class=\"card success\">\r\n";
                message += "\t\t\t\t\t<div class=\"card__content\">\r\n";
                message += "\t\t\t\t\t\t<h3>Passed</h3>\r\n";
                message += "\t\t\t\t\t\t<div class=\"counter\">\r\n";
                message += "\t\t\t\t\t\t\t<div>" + passedInt + "</div>\r\n";
                message += "\t\t\t\t\t\t\t<svg class=\"check\">\r\n";
                message +=
                    "\t\t\t\t\t\t\t\t<line x1=\"5\" y1=\"15\" x2=\"15\" y2=\"25\" />\r\n";
                message +=
                    "\t\t\t\t\t\t\t\t<line x1=\"13\" y1=\"25\" x2=\"30\" y2=\"5\" />\r\n";
                message += "\t\t\t\t\t\t\t<svg>\r\n";
                message += "\t\t\t\t\t\t</div>\r\n";
                message += "\t\t\t\t\t</div>\r\n";
                message += "\t\t\t\t</div>\r\n";
                message += "\t\t\t</div>\r\n";

                // Failed
                message += "\t\t\t<div class=\"col-4\">\r\n";
                message += "\t\t\t\t<div class=\"card error\">\r\n";
                message += "\t\t\t\t\t<div class=\"card__content\">\r\n";
                message += "\t\t\t\t\t\t<h3>Failed</h3>\r\n";
                message += "\t\t\t\t\t\t<div class=\"counter\">\r\n";
                message += "\t\t\t\t\t\t\t<div>" + failedInt + "</div>\r\n";
                message += "\t\t\t\t\t\t\t<svg class=\"cross\">\r\n";
                message +=
                    "\t\t\t\t\t\t\t\t<line x1=\"1\" y1=\"25\" x2=\"25\" y2=\"1\" />\r\n";
                message +=
                    "\t\t\t\t\t\t\t\t<line x1=\"1\" y1=\"1\" x2=\"25\" y2=\"25\" />\r\n";
                message += "\t\t\t\t\t\t\t<svg>\r\n";
                message += "\t\t\t\t\t\t</div>\r\n";
                message += "\t\t\t\t\t</div>\r\n";
                message += "\t\t\t\t</div>\r\n";
                message += "\t\t\t</div>\r\n";
```

```
            message += "\t\t</div>\r\n";

            // Header details
            message += "\t\t<div class=\"list\">\r\n";
            message += "\t\t\t<div class=\"list__content\">\r\n";
            message += "\t\t\t\t<div class=\"list__content__row\">\r\n";
            message += "\t\t\t\t\t<div class=\"row details\">\r\n";
            message += "\t\t\t\t\t\t<div class=\"col-1\">\r\n";
            message +=
                "\t\t\t\t\t\t\t<span><strong>Nr.</strong></span>\r\n";
            message += "\t\t\t\t\t\t</div>\r\n";
            message += "\t\t\t\t\t\t<div class=\"col-3 action\">\r\n";
            message +=
                "\t\t\t\t\t\t\t<span><strong>attribute</strong></span>\r\n";
            message += "\t\t\t\t\t\t</div>\r\n";
            message += "\t\t\t\t\t</div>\r\n";
            message +=
                "\t\t\t\t\t<div class=\"element\"><strong>element name</strong>\r\n";
            message += "\t\t\t\t</div>\r\n";
            message += "\t\t\t</div>\r\n";

            var showErrorDisplay =
                OverallSettings.ShowErrorDisplay.ToString();

            // Details
            foreach (var testResult in testResults)
            {
                var testOK = "list__content__row success";
                if (testResult.result == "4")
                    testOK = "list__content__row error";

                if (showErrorDisplay == "0" && testResult.result == "4")
                {
                    message += "\t\t\t\t<div class=\"" + testOK + "\">\r\n";
                    message += "\t\t\t\t\t<div class=\"row details\">\r\n";
                    message += "\t\t\t\t\t\t<div class=\"col-1\">\r\n";
                    message += "\t\t\t\t\t\t\t<span>" + testResult.testnr +
                                   "</span>\r\n";
                    message += "\t\t\t\t\t\t</div>\r\n";
                    message +=
                        "\t\t\t\t\t\t<div class=\"col-3 action\">\r\n";
                    message += "\t\t\t\t\t\t\t<span>" +
                                   testResult.attribute + "</span>\r\n";
                    message += "\t\t\t\t\t\t</div>\r\n";
                    message += "\t\t\t\t\t</div>\r\n";
                    message +=
                        "\t\t\t\t\t<div class=\"element\">" +
                        testResult.elementname + "(" +
                        testResult.element + ")</div>\r\n";
                    message += "\t\t\t\t</div>\r\n";
                }
```

```
                    if (showErrorDisplay == "1")
                    {
                        message += "\t\t\t\t<div class=\"" + testOK + "\">\r\n";
                        message += "\t\t\t\t\t<div class=\"row details\">\r\n";
                        message += "\t\t\t\t\t\t<div class=\"col-1\">\r\n";
                        message += "\t\t\t\t\t\t\t<span>" + testResult.testnr +
                                   "</span>\r\n";
                        message += "\t\t\t\t\t\t</div>\r\n";
                        message +=
                            "\t\t\t\t\t\t<div class=\"col-3 action\">\r\n";
                        message += "\t\t\t\t\t\t\t<span>" +
                                   testResult.attribute + "</span>\r\n";
                        message += "\t\t\t\t\t\t</div>\r\n";
                        message += "\t\t\t\t\t</div>\r\n";
                        message +=
                            "\t\t\t\t\t<div class=\"element\">" +
                            testResult.elementname + "(" +
                            testResult.element + ")</div>\r\n";
                        message += "\t\t\t\t</div>\r\n";
                    }
                }

                message += "\t\t\t</div>\r\n";
                message += "\t\t</div>\r\n";
                message += "\t</div>\r\n";
                message += "</body>\r\n";
                message += "</html>";
                TestReportToFile(message, timestring);
            }
            catch (Exception e)
            {
                Console.WriteLine(e);
                MessageBox.Show(e.Message);
            }
        }

        /// <summary>
        ///     Deteles the images.
        /// </summary>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for DeteleImages
        public static void DeteleImages()
        {
            var screenshotsPath = GetCurrentDir(1);
            var di = new DirectoryInfo(screenshotsPath + "screenshots\\" +
                                       applicationName.Trim());
            foreach (var file in di.GetFiles()) file.Delete();
        }

        /// <summary>
        ///     Gets the counter check.
        /// </summary>
        /// <returns>System.Int32.</returns>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for GetCounterCheck
        public static int GetCounterCheck()
        {
            return counterCheck;
        }

        /// <summary>
        ///     Gets the credentials.
        /// </summary>
        /// <param name="bestandsnaam">The bestandsnaam.</param>
        /// <returns>Credentials.</returns>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for GetCredentials
        public static Credentials GetCredentials(string bestandsnaam)
        {
            var credentials = new Credentials(bestandsnaam);
            return credentials;
        }

        /// <summary>
        ///     Gets the current dir.
        /// </summary>
        /// <param name="niveau">The niveau.</param>
        /// <returns>System.String.</returns>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for GetCurrentDir
        public static string GetCurrentDir(int niveau)
        {
            var ValueList = new List<string>();
            if (File.Exists("settings.xml"))
            {
                var xDoc = new XmlDocument();
                xDoc.Load("settings.xml");
                try
                {
                    var xelement = XElement.Load("settings.xml");
                    var elementVars = xelement.Elements();
                    foreach (var elementVar in elementVars)
                        ValueList.Add(elementVar.Value);
                }
                catch (Exception ex)
                {
                    General.LogMessage(ex.Message + ex.StackTrace, 4);
                    MessageBox.Show(
                        "The 'settings.xml' file is not found!!!\r\nPlease fill in the correct drive and directory.\r\n"
                        + Environment.CurrentDirectory,
                        "Settings Error",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
                    ValueList[0] = Environment.CurrentDirectory;
                }
            }
            else
            {
                MessageBox.Show(
                    "The 'settings.xml' file is not found!!!\r\nPlease fill in the correct drive and directory.\r\n"
                    + Environment.CurrentDirectory,
                    "Settings Error",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
            }

            return ValueList[niveau];
        }

        /// <summary>
        ///     Gets the directory.
        /// </summary>
        /// <returns>System.String.</returns>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for GetDirectory
        public static string GetDirectory()
        {
            var driverPath = Environment.CurrentDirectory;

            var directory = driverPath.Split('\\');
            var intX = 0;
            foreach (var dir in directory)
                if (dir == "SeleniumCSharpTests")
                    break;
                else
                    intX++;

            var di = string.Empty;
            for (var x = 0; x < directory.Length - intX - 1; x++)
                di += directory[x] + "\\";

            return di;
        }

        /// <summary>
        ///     Gets the settings XML strings.
        /// </summary>
        /// <param name="xmlFile">The XML file.</param>
        /// <param name="dirnr">The dirnr.</param>
        /// <returns>List&lt;System.String&gt;.</returns>
        public static List<string> GetSettingsXmlStrings(string xmlFile,
            int dirnr = 0)
        {
            var ValueList = new List<string>();

            var di = GetCurrentDir(dirnr);
            if (File.Exists(di + xmlFile + ".xml"))
            {
                var xDoc = new XmlDocument();
                xDoc.Load(di + xmlFile + ".xml");
                try
                {
                    var xelement = XElement.Load(di + xmlFile + ".xml");
                    var elementVars = xelement.Elements().Elements();
                    foreach (var elementVar in elementVars)
                        ValueList.Add(elementVar.Value);
                }
                catch (Exception ex)
                {
                    General.LogMessage(ex.Message + ex.StackTrace, 4);
                    MessageBox.Show(
                        "The '" + di + xmlFile + ".xml"
                        + ".xml' file is not found!!!\r\nPlease fill in the correct drive and directory.",
                        "Settings Error",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
                }
            }
            else
            {
                if (xmlFile != "serialXML")
                    MessageBox.Show(
                        "The '" + di + xmlFile
                        + ".xml' file is not found!!!\r\nPlease fill in the correct drive and directory.",
                        "Settings Error",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
            }

            return ValueList;
        }

        /// <summary>
        ///     Haals the elementop.
        /// </summary>
        /// <param name="driver">The driver.</param>
        /// <param name="testElement">The test element.</param>
        /// <param name="testAttribute">The test attribute.</param>
        /// <returns>IWebElement.</returns>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for HaalElementop
        public static IWebElement HaalElementop(IWebDriver driver,
            string testElement, string testAttribute)
        {
            IWebElement webElement = null;

            switch (testAttribute)
            {
                case "ID":
                    // select the drop down list
                    webElement = driver.FindElement(By.Id(testElement));
                    break;
                case "NAME":
                    webElement = driver.FindElement(By.Name(testElement));
                    break;
                case "CLASS":
                    webElement = driver.FindElement(By.ClassName(testElement));
                    break;
                case "XPATH":
                    webElement = driver.FindElement(By.XPath(testElement));
                    break;
                case "CSSSELECTOR":
                    webElement =
                        driver.FindElement(By.CssSelector(testElement));
                    break;
            }

            return webElement;
        }

        public static bool IsElementVisible(IWebElement element)
        {
            return element.Displayed && element.Enabled;
        }

        /// <summary>
        ///     Highlights the and screenshot.
        /// </summary>
        /// <param name="driver1">The driver1.</param>
        /// <param name="context">The context.</param>
        /// <param name="sleepTime">The sleep time.</param>
        /// <param name="borderColor">Color of the border.</param>
        /// <param name="borderWidth">Width of the border.</param>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for HighlightAndScreenshot
        public static void HighlightAndScreenshot(
            IWebDriver driver1,
            IWebElement context,
            int sleepTime,
            string borderColor,
            int borderWidth)
        {
            var oldScript = context.GetAttribute("style");
            var rc = (RemoteWebElement)context;
            var actions = new Actions(driver1);
            if (rc.Displayed)
            {
                var visible = IsElementVisible(context);
                if (visible)
                {
                    actions = new Actions(driver1);
                    actions.MoveToElement(context);
                    actions.Perform();
                }

                var driver = (IJavaScriptExecutor)rc.WrappedDriver;

                var script = "arguments[0].style.cssText = \"border-width: " +
                             borderWidth
                             + "px; border-style: solid; border-color: "
                             + borderColor.ToLower() + "\";";
                driver.ExecuteScript(script, rc);

                Thread.Sleep(sleepTime);

                var clear = string.Empty;
                if (oldScript.Length > 0)
                    clear = @"arguments[0].style.cssText = 'oldScript'";
                else
                    clear = @"arguments[0].style.cssText = """"; ";

                driver.ExecuteScript(clear, rc);
            }
        }

        /// <summary>
        ///     Initalizes the serial key.
        /// </summary>
        /// <returns>System.Int32.</returns>
        public static int InitalizeSerialKey()
        {
            var value = 1;
            var xmlSerial = GetSettingsXmlStrings("serialXML");
            if (xmlSerial.Count > 0)
            {
                serialKeyXml = new SerialKeyXML(
                    xmlSerial[0],
                    xmlSerial[1],
                    xmlSerial[2],
                    xmlSerial[3],
                    xmlSerial[4],
                    xmlSerial[5]);
                General.SetSerialKeyXML(serialKeyXml);
                value = 2;
            }

            return value;
        }

        /// <summary>
        ///     Initializes the database connection.
        /// </summary>
        /// <param name="loginU">if set to <c>true</c> [login u].</param>
        public static void InitializeDatabaseConnection(bool loginU)
        {
            /*
             * Gebruiker dient in te loggen op een andere database als waar de data staan
             *
             */
            //var host = "SCHMEITSNAS.SYNOLOGY.ME";
            //var database = "autotest";
            //var port = "3306";
            //var user = "autotest";
            //var password = "NiFSi1fZjbJ4zXE8";
            //
            var host = "";
            var database = "";
            var port = "";
            var user = "";
            var password = "";
            General.LogMessage("InitializeDatabaseConnection(" + loginU + ");",
                1);

            if (loginU == false)
            {
                var xelement = XElement.Load(GetCurrentDir(0) + "connect.xml");
                General.LogMessage("xelement: " + xelement, 1);
                var testsElements = xelement.Elements();
                foreach (var testElement in testsElements)
                {
                    host = testElement.Element("host").Value;
                    database = testElement.Element("database").Value;
                    port = testElement.Element("port").Value;
                    user = testElement.Element("username").Value;
                    password = testElement.Element("password").Value;
                }
            }

            // anders hard gecodeerde gegevens gebruiker voor de database connectie.
            databaseConnection = new DatabaseConnection(
                host,
                database,
                port,
                user,
                password);

            General.LogMessage(databaseConnection.ToString(), 1);
            General.LogMessage("Host:     " + databaseConnection.Host, 1);
            General.LogMessage("Database: " + databaseConnection.Database, 1);
            General.LogMessage("Port:     " + databaseConnection.Port, 1);
            General.LogMessage("User:     " + databaseConnection.User, 1);
            General.LogMessage("Password: " + databaseConnection.Password, 1);
            General.SetDatabaseConnection(databaseConnection);
        }

        /// <summary>
        ///     Determines whether [is element present] [the specified driver].
        /// </summary>
        /// <param name="driver">The driver.</param>
        /// <param name="idString">The identifier string.</param>
        /// <returns><c>true</c> if [is element present] [the specified driver]; otherwise, <c>false</c>.</returns>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for IsElementPresent
        public static bool IsElementPresent(IWebDriver driver, string
         idString)
        {
            try
            {
                driver.FindElement(By.Id(idString));
                return true;
            }
            catch (NoSuchElementException)
            {
                return false;
            }
        }

        /// <summary>
        ///     Determines whether this instance is windows10.
        /// </summary>
        /// <returns><c>true</c> if this instance is windows10; otherwise,
          <c>false</c>.</returns>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for IsWindows10
        public static bool IsWindows10()
        {
            var reg =
                Registry.LocalMachine.OpenSubKey(
                    @"SOFTWARE\Microsoft\Windows NT\CurrentVersion");

            var productName = (string)reg.GetValue("ProductName");

            return productName.StartsWith("Windows 10");
        }

        public static void TestUitgevoerd(bool uit)
        {
            uitgevoerd = uit;
        }

        /// <summary>
        ///     Sets the name of the application.
        /// </summary>
        /// <param name="name">The name.</param>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for setApplicationName
        public static void setApplicationName(string name)
        {
            applicationName = name;
        }

        /// <summary>
        ///     Sets the name of the class.
        /// </summary>
        /// <param name="name">The name.</param>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for setClassName
        public static void setClassName(string name)
        {
            className = name;
        }

        /// <summary>
        ///     Sets the counter check.
        /// </summary>
        /// <param name="value">The value.</param>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for SetCounterCheck
        public static void SetCounterCheck(int value)
        {
            counterCheck = value;
        }

        /// <summary>
        ///     Sets the HTML code.
        /// </summary>
        /// <param name="timestring">The timestring.</param>
        /// <returns>System.String.</returns>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for SetHTMLCode
        public static string SetHTMLCode(string timestring)
        {
            var testResults = new List<TestResults.TestResultsHTML>();
            testResults = TestResults.GetTestResultsHTML(timestring);
            setApplicationName(testResults[0].application);

            var screenshotsPath = GetCurrentDir(1);
            var pathG = screenshotsPath.Split('\\');
            var g = string.Empty;

            for (var i = 0; i < pathG.Length - 2; i++) g += pathG[i] + "\\";

            screenshotsPath = g;
            var name1 = applicationName.Trim();

            CheckScreenshotDir("TestReports\\" + name1);

            var relativePath = screenshotsPath + "TestReports\\" + name1 + "\\";
            HtmlPage = relativePath + "TestReport_" + timestring + ".html";
            return HtmlPage;
        }

        /// <summary>
        ///     Sets the testnr.
        /// </summary>
        /// <param name="testnr">The testnr.</param>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for setTestnr
        public static void setTestnr(string testnr)
        {
            Functions.testnr = testnr;
        }

        /// <summary>
        ///     Sets the testrun identifier.
        /// </summary>
        /// <param name="testrun">The testrun.</param>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for setTestrunID
        public static void setTestrunID(string testrun)
        {
            _testrun = testrun;
        }

        public static void setProjectName(string project)
        {
            _project = project;
        }

        public static int getProjectID()
        {
            var projectID = 0;
            if (_project != null)
            {
                var query =
                    "SELECT id FROM projects WHERE UPPER(project_name) = '" +
                    _project.ToUpper() + "';";
                var ds = GenericDataRead.GetData(query);
                if (ds.Rows.Count > 0)
                    projectID = Convert.ToInt32(ds.Rows[0][0].ToString());
            }

            return projectID;
        }

        /// <summary>
        ///     Shows the data.
        /// </summary>
        /// <param name="driver">The driver.</param>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for ShowData
        public static void ShowData(IWebDriver driver)
        {
            var PageURL = driver.Url;
            var PageSource = driver.PageSource;

            var stringL = PageSource;
            General.LogMessage(stringL, 5);
        }

        /// <summary>
        ///     Splits the bestand.
        /// </summary>
        /// <param name="bestand">The bestand.</param>
        /// <returns>System.String.</returns>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for SplitBestand
        public static string SplitBestand(string bestand)
        {
            var ele = bestand.Split('\\');
            var laatste = ele[ele.Length - 1];
            var naam = laatste.Split('.');
            return naam[0];
        }

        /// <summary>
        ///     Switches to next tab.
        /// </summary>
        /// <param name="driver">The driver.</param>
        /// <param name="handlesNr">The handles nr.</param>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for SwitchToNextTab
        public static void SwitchToNextTab(IWebDriver driver, int handlesNr)
        {
            if (driver.WindowHandles.Count > handlesNr)
            {
                driver.SwitchTo().Window(driver.WindowHandles[handlesNr]);
            }
            else
            {
                Thread.Sleep(2000);
                SwitchToNextTab(driver, handlesNr);
            }
        }

        /// <summary>
        ///     Takes the screenshot.
        /// </summary>
        /// <param name="driver">The driver.</param>
        /// <param name="result">The result.</param>
        /// <param name="name">The name.</param>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for TakeScreenshot
        public static void TakeScreenshot(IWebDriver driver, int result,
            string name = "screenshot")
        {
            var screenshotsPath = GetCurrentDir(1);
            var pathG = screenshotsPath.Split('\\');
            var g = string.Empty;

            for (var i = 0; i < pathG.Length - 2; i++) g += pathG[i] + "\\";

            screenshotsPath = g;
            var name1 = applicationName.Trim();

            if (result == 4)
            {
                var relativePath =
                    screenshotsPath + "screenshots\\" + name1 + "\\";
                ScreenshotName =
                    relativePath + name1 + "-" +
                    string.Format("{0:yyyyMMddHHmmssFFF}", DateTime.UtcNow)
                    + "testnr_" + SetTestnr + ".png";
                General.LogMessage("Screenshot name: " + ScreenshotName, 1);
                driver.TakeScreenshot().SaveAsFile(ScreenshotName,
                    ScreenshotImageFormat.Png);

                General.LogMessage("Error", result, name1,
                    Convert.ToInt32(SetTestnr));
            }

            ShowData(driver);
        }

        /// <summary>
        ///     Teksts the vervanging.
        /// </summary>
        /// <param name="tekst">The tekst.</param>
        /// <returns>System.String.</returns>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for TekstVervanging
        public static string TekstVervanging(string tekst)
        {
            var tekstGewijzigd = tekst;
            switch (tekst)
            {
                case "=CurrentDate()":
                    tekstGewijzigd = DateTime.Now.Day + "-" +
                                     DateTime.Now.Month + "-" +
                                     DateTime.Now.Year;
                    break;
            }

            return tekstGewijzigd;
        }

        /// <summary>
        ///     Tests the report to file.
        /// </summary>
        /// <param name="message">The message.</param>
        /// <param name="timestring">The timestring.</param>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for TestReportToFile
        public static void TestReportToFile(string message, string timestring)
        {
            SetHTMLCode(Convert.ToInt32(timestring.Substring(4, 8)).ToString());
            var sw = File.AppendText(HtmlPage); // Change filename
            try
            {
                var logLine = $"{message}";
                sw.WriteLine(logLine);
            }
            finally
            {
                sw.Close();
            }
        }

        public static void CodeToFile(string message, string timestring)
        {
            var sw = File.AppendText(HtmlPage); // Change filename
            try
            {
                var logLine = $"{message}";
                sw.WriteLine(logLine);
            }
            finally
            {
                sw.Close();
            }
        }

        /// <summary>
        ///     Teststaps the specified driver.
        /// </summary>
        /// <param name="driver">The driver.</param>
        /// <param name="machinestatic">The machinestatic.</param>
        /// <param name="radiobutton">The radiobutton.</param>
        /// <param name="version">The version.</param>
        /// <param name="bestandsnaamArgument">The bestandsnaam argument.</param>
        /// <autogeneratedoc />
        /// TODO Edit XML Comment Template for Teststap
        public static void Teststap(IWebDriver driver, string machinestatic,
            string radiobutton, string version,
            string bestandsnaamArgument, bool opslaan, string saveWat = "")
        {
            var table = General.GetTestrun(bestandsnaamArgument);
            var actions = new Actions(driver);
            // var credits = GetCredentials(bestandsnaamArgument);

            var inhoud = "";
            var vestandsnaam = "";
            var credits =
                GetCredentials(_project + @"\" + bestandsnaamArgument);

            var machineNumber = Convert.ToString(getMachineCode());

            for (var intx = 0; intx < table.Rows.Count; intx++)
            {
                var strikt = true;
                var savedAttribute =
                    table.Rows[intx]["save_attribute"].ToString().Trim();
                var drTitle = driver.Title;
                if (drTitle != null)
                {
                    vestandsnaam =
                        table.Rows[intx]["testcase"].ToString().Trim();
                    setClassName(table.Rows[intx]["testcase"].ToString()
                        .Trim());
                    setTestnr(table.Rows[intx]["testnr"].ToString().Trim());

                    if (table.Rows[intx]["test_check_strict"].ToString()
                        .ToUpper() == "TRUE")
                        strikt = true;
                    else
                        strikt = false;
```

```

var v = (table != null
    ? table != null ? table.Rows[intx]["testurl"] : null
    : null) != string.Empty;

if (table.Rows[intx]["testurl"] != null &&
    table.Rows.Count > intx && v)
    if (table.Rows[intx]["testurl"].ToString().Trim()
            .Length > 0
        && table.Rows[intx]["testswitch"].ToString()
            .ToUpper() == "YES")
    {
        var handles = driver.WindowHandles;
        foreach (var handle in handles)
        {
            var contains = driver.SwitchTo().Window(handle)
                .Url
                .Contains(table.Rows[intx]["testurl"]
                    .ToString().Trim());
        }
    }

try
{
    if (table.Rows[intx]["testaction"].ToString() == "wait")
    {
        Thread.Sleep(Convert.ToInt32(
            table.Rows[intx]["testtext"].ToString()
                .Trim()) * 1000);
    }
    else
    {
        var idString = table.Rows[intx]["testelement"]
            .ToString().Trim();
        var testname = table.Rows[intx]["testname"]
            .ToString().Trim();
        var testcase = table.Rows[intx]["testcase"]
            .ToString().Trim();
        var testaction = table.Rows[intx]["testaction"]
            .ToString().Trim();

        if(idString != string.Empty)
        {
            Assert.IsTrue(
                CheckBy(
                    driver,
                    table.Rows[intx]["testelement"].ToString()
                        .Trim(),
                    table.Rows[intx]["testattribute"].ToString()
                        .Trim(),
                    table.Rows[intx]["testnr"].ToString()
                        .Trim(),
                    radiobutton,
                    table.Rows[intx]["testext_check"].ToString()
                        .Trim(),
                    table.Rows[intx]["testaction"].ToString()
                        .Trim(),
                    table.Rows[intx]["testurl"].ToString()
                        .Trim(),
                    table.Rows[intx]["testswitch"].ToString()
                        .Trim(),
                    table.Rows[intx]["testdescription"]
                        .ToString().Trim(),
                    table.Rows[intx]["testinverse"].ToString()
                        .Trim(),
                    table.Rows[intx]["test_comment"].ToString()
                        .Trim(),
                    table.Rows[intx]["testname"].ToString()
                        .Trim(),
                    table.Rows[intx]["testcase"].ToString()
                        .Trim(),
                    version,
                    0,
                    machineNumber,
                    table.Rows[intx]["testlogicalobjectname"]
                        .ToString().Trim(),
                    credits.Url,
                    table.Rows[intx]["testtext"].ToString()
                        .Trim(),
                    bestandsnaamArgument,
                    "",
                    opslaan,
                    strikt
                ));
        }


        IWebElement content = null;
        var tekst1 = string.Empty;

        var testElement = table.Rows[intx]["testelement"]
            .ToString().Trim();




        var testAttribute =
            table.Rows[intx]["testattribute"].ToString()
```

```
                                    .Trim()
                                    .ToUpper();
                            if (testElement != string.Empty)
                            {
                                switch (testAttribute)
                                {
                                    case "ID":
                                        content =
                                            driver.FindElement(By.Id(testElement));
                                        break;
                                    case "NAME":
                                        content =
                                            driver.FindElement(
                                                By.Name(testElement));
                                        break;
                                    case "CLASS":
                                        content =
                                            driver.FindElement(
                                                By.ClassName(testElement));
                                        break;
                                    case "XPATH":
                                        content =
                                            driver.FindElement(
                                                By.XPath(testElement));
                                        break;
                                    case "CSSSELECTOR":
                                        content =
                                            driver.FindElement(
                                                By.CssSelector(testElement));
                                        break;
                                }
                            }

                            if (table.Rows[intx]["testelement"].ToString()
                                    .Trim().Length == 0
                                || table.Rows[intx]["testelement"].ToString()
                                    .ToUpper().Trim() == "POPUP")
                            {
                                if (table.Rows[intx]["testtext"].ToString()
                                    .ToUpper() == "DISMISS")
                                    ExpectedConditions.AlertIsPresent()
                                        .Invoke(driver).Dismiss();
                                if (table.Rows[intx]["testtext"].ToString()
                                    .ToUpper() == "ACCEPT")
                                    ExpectedConditions.AlertIsPresent()
                                        .Invoke(driver).Accept();
                            }


                            if (table.Rows[intx]["testaction"].ToString() !=
                                null)
                                if (table.Rows[intx]["testaction"].ToString()
                                        .Trim().Length > 0
                                    || table.Rows[intx]["testaction"].ToString()
```

```
  .Trim().ToLower() != "empty")
switch (table.Rows[intx]["testaction"]
  .ToString())
{
  case "switch to url":
    driver.Navigate()
      .GoToUrl(
        table.Rows[intx]["testurl"]
          .ToString());

    inhoud +=
      "driver.Navigate().GoToUrl('" +
      table.Rows[intx]["testurl"] +
      "')\r\n";

    break;
  case "click":
    actions = new Actions(driver);
    actions.MoveToElement(content);
    actions.Perform();
    content.Click();
    break;
  case "value":
    if (table.Rows[intx]["testtext"] !=
      null && table.Rows.Count > intx
      && (string)table.Rows[intx][
        "testtext"]
      != string.Empty)
    {
      tekst1 = TekstVervanging(
        table.Rows[intx]["testtext"]
          .ToString().Trim());
      actions = new Actions(driver);
      actions.MoveToElement(content);
      actions.Perform();
      content.Click();
      content.Clear();
      content.SendKeys(tekst1);
    }

    break;
  case "sendkeys":
    if (table.Rows[intx]["testtext"] !=
      null && table.Rows.Count > intx
      && (string)table.Rows[intx][
        "testtext"]
      != string.Empty)
```

```
                                                tekst1 = TekstVervanging(
                                                    table.Rows[intx]["testtext"]
                                                        .ToString().Trim());
                                            if (table.Rows[intx][
                                                    "test_password"] != null &&
                                                table.Rows.Count > intx
                                                && table.Rows[intx][
                                                    "test_password"].ToString()
                                                != string.Empty)
                                                tekst1 = TekstVervanging(
                                                    table.Rows[intx][
                                                            "test_password"]
                                                        .ToString().Trim());
                                            actions = new Actions(driver);

                                            actions.MoveToElement(content);
                                            actions.Perform();

                                            var tekst27 = tekst1;

                                            if (tekst1.ToUpper()
                                                .Contains("{ENTER}"))
                                            {
                                                var splitTekst =
                                                    tekst1.Split('{');
                                                content.Click();
                                                content.Clear();
                                                content.SendKeys(splitTekst[0] +
                                                    Keys.Enter);
                                            }
                                            else
                                            {
                                                switch (tekst1.ToUpper())
                                                {
                                                    case "TAB":
                                                        tekst1 = Keys.Tab;
                                                        break;
                                                    case "PGUP":
                                                        tekst1 = Keys.PageUp;
                                                        break;
                                                    case "PGDN":
                                                        tekst1 = Keys.PageDown;
                                                        break;
                                                    case "ENTER":
                                                        tekst1 = Keys.Enter;
                                                        break;
                                                    case "ESC":
                                                        tekst1 = Keys.Escape;
                                                        break;
                                                }

```

```
        content.Click();
        if (tekst27.ToUpper() !=
            "ENTER" &&
            tekst27.ToUpper() !=
            "TAB" &&
            tekst27.ToUpper() !=
            "PGUP" &&
            tekst27.ToUpper() !=
            "PGDN" &&
            tekst27.ToUpper() != "ESC")
            content.Clear();
        content.SendKeys(tekst1);
    }


    break;

case "select":
    var tekst2 = TekstVervanging(
        table.Rows[intx]["testtext"]
            .ToString().Trim());

    IWebElement education = null;
    SelectElement selectElement = null;

    switch (table.Rows[intx][
            "testattribute"].ToString()
        .Trim().ToUpper())
    {
        case "ID":
            // select the drop down list
            education =
                driver.FindElement(
                    By.Id(table.Rows[
                            intx][
                            "testelement"]
                        .ToString()
                        .Trim()));
            break;
        case "NAME":
            education =
                driver.FindElement(
                    By.Name(
                        table.Rows[intx]
                            ["testelement"]
                            .ToString()
                            .Trim()));
```

```
                          break;
                        case "CLASS":
                          education =
                            driver.FindElement(
                              By.ClassName(
                                table.Rows
          [intx]

["testelement"]
                                  .ToString
          ()
                                  .Trim
          ()));
                          break;
                        case "XPATH":
                          education =
                            driver.FindElement(
                              By.XPath(
                                table.Rows
          [intx]

["testelement"]
                                  .ToString
          ()
                                  .Trim
          ()));
                          break;
                        case "CSSSELECTOR":
                          education =
                            driver.FindElement(
                              By.CssSelector(
                                table.Rows
          [intx]

["testelement"]
                                  .ToString
          ()
                                  .Trim
          ()));
                          break;
                      }

                      //create select element object
                      selectElement =
                        new SelectElement(education);
                      // select by text
                      selectElement.SelectByText
          (tekst2);
                      break;
                    case "move_to":
                      actions = new Actions(driver);
                      actions.MoveToElement(content);
                      actions.Perform();
                      break;

                    case "upload":
```

```
                                            tekst1 = TekstVervanging(
                                                table.Rows[intx]["testtext"]
                                                    .ToString().Trim());
                                            var file1 =
                                                tekst1.Replace("\'", "\\'");
                                            content.Click();
                                            Thread.Sleep(5000);
                                            SendKeys.SendWait(file1);
                                            SendKeys.SendWait(@"{Enter}");
                                            break;
                                        case "checkbox":
                                            IWebElement checkbox1 = null;

                                            checkbox1 = HaalElementop(driver,
                                                testElement, testAttribute);

                                            var oldScript =
                                                checkbox1.GetAttribute(
                                                    "checked");

                                            var tekst3 =
                                                table.Rows[intx]["testtext"]
                                                    .ToString().Trim();
                                            var rc =
                                                (RemoteWebElement)checkbox1;
                                            var driver1 =
                                                (IJavaScriptExecutor)rc
                                                    .WrappedDriver;
                                            var script1 = "";
                                            if (tekst3.ToUpper() == "TRUE" &&
                                                oldScript == null)
                                                script1 =
                                                    "arguments[0].setAttribute('checked', 'true');";
                                            if (tekst3.ToUpper() == "FALSE" &&
                                                oldScript != null)
                                                script1 =
                                                    "arguments[0].removeAttribute('checked');";

                                            driver1.ExecuteScript(script1, rc);
                                            break;
                                        case "switch to iFrame":
                                            var tekstFrame =
                                                table.Rows[intx]["testtext"]
                                                    .ToString().Trim();
                                            var detailFrame = content;
                                            driver.SwitchTo()
                                                .Frame(detailFrame);
                                            break;
                                        case "switch to default":
                                            driver.SwitchTo().DefaultContent();
                                            break;
```

```
case "set_value":
    var query = string.Empty;

    query =
        "SELECT * FROM saved_values WHERE ";
    query +=
        "saved_values_testname = '" +
        bestandsnaamArgument + "' AND ";
    query += "saved_values_name = '" +
        table.Rows[intx]["testtext"]
            .ToString().Trim() + "';";
    var dt =
        GenericDataRead.GetData(query);

    if (dt.Rows.Count > 0)
    {
        // Indien variable voor testscript bestaat, update deze uit tabel 'saved_values'
        var save_id = dt.Rows[0][0]
            .ToString();
        query =
            "UPDATE saved_values SET saved_values_value = '";
        if (savedAttribute == string.Empty)
            query += content.Text.Trim()
                    .Replace("\r\n",
                        string.Empty) +
                "' ";
        if (savedAttribute.ToUpper() == "HREF")
            query += content.GetAttribute(
                    "href").Trim()
                +
                "' ";
        if (savedAttribute.ToUpper() == "ID")
            query += content.GetAttribute(
                         "id").Trim()
                     +
                     "' ";
        if (savedAttribute.ToUpper() == "NAME")
            query += content.GetAttribute(
                         "name").Trim()
                     +
                     "' ";
              query += "WHERE ";
              query += "saved_values_id = " +
                save_id + ";";
            }
            else
            {
              // anders Voeg de gegevens toe aan de tabel 'saved_values'
              query =
                "INSERT INTO saved_values (saved_values_testname, ";
              query +=
                "saved_values_name, saved_values_value, projectid) ";
              query += "VALUES('" +
                bestandsnaamArgument +
                "', ";
              query += "'" +
                table.Rows[intx]["testtext"]
                  .ToString().Trim() +
                "', '";

              if (savedAttribute == string.Empty)
                query += content.Text.Trim()
                    .Replace("\r\n",
                      string.Empty) +
                  "', ";
              if (savedAttribute.ToUpper() == "HREF")
                query += content.GetAttribute(
                    "href").Trim()
                  +
                  "', ";
              if (savedAttribute.ToUpper() == "ID")
                query += content.GetAttribute(
                       "id").Trim()
                     +
                     "', ";
              if (savedAttribute.ToUpper() == "NAME")
                query += content.GetAttribute(
                       "name").Trim()
                     +
                     "', ";
```

```
                                query += getProjectID() + ");";
                            }

                            GenericDataRead.INUPDEL(query);
                            break;
                        case "get_value":

                            query =
                                "SELECT saved_values_value, attribute FROM saved_values ";
                            // query += "WHERE saved_values_testname = '" + bestandsnaamArgument + "' AND ";
                            query += "WHERE ";
                            query += "projectid = " +
                                getProjectID() + " AND ";
                            query += "saved_values_name = '" +
                                table.Rows[intx]["testtext"]
                                    .ToString().Trim() + "';";
                            var dtget =
                                GenericDataRead.GetData(query);

                            if (dtget.Rows.Count > 0)
                            {
                                if (dtget.Rows[0][1]
                                    .ToString() == string.Empty)
                                {
                                    actions =
                                        new Actions(driver);
                                    actions.MoveToElement(
                                        content);
                                    actions.Perform();
                                    var tekstget =
                                        dtget.Rows[0][0]
                                            .ToString();
                                    content.Click();
                                    content.Clear();
                                    content.SendKeys(tekstget);
                                }

                                if (dtget.Rows[0][1]
                                    .ToString().ToUpper() == "HREF")
                                {
                                    driver.Navigate()
                                        .GoToUrl(
                                            dtget.Rows[0][0]
                                                .ToString());
                                }
                                if (dtget.Rows[0][1]
                                        .ToString().ToUpper() == "ID")
                                {
                                    MessageBox.Show("Waarde ID: " + dtget.Rows[0][0].ToString());
                                }
                                if (dtget.Rows[0][1]
                                        .ToString().ToUpper() == "NAME")
                                {
                                    MessageBox.Show("Waarde Name: " + dtget.Rows[0][0].ToString());
                                }
                            }

                            break;
                        case "scroll":
                            var richting =
                                table.Rows[intx]["testtext"]
                                    .ToString().Trim();
                            var js =
                                (IJavaScriptExecutor)driver;
                            if (richting.ToUpper() == "BOTTOM")
                                js.ExecuteScript(
                                    "window.scrollTo(0, document.body.scrollHeight)");
                            if (richting.ToUpper() == "TOP")
                                js.ExecuteScript(
                                    "window.scrollTo(0, 0)");
                            if (richting.ToUpper() == "DOWN")
                                js.ExecuteScript(
                                    "window.scrollTo(0, 1000)");
                            if (richting.ToUpper() == "UP")
                                js.ExecuteScript(
                                    "window.scrollTo(0, 1000)");
                            break;
                        case "cmd":
                            if (intx != 0)
                                using (var cmd =
                                    Process.Start(
                                        table.Rows[intx][
                                                "testtext"]
                                            .ToString().Trim()))
                                {
                                    cmd.WaitForExit();
                                }

                            break;
                        case "screenshot":
                            var screenshotsPath =
                                GetCurrentDir(1);
                            var pathG =
                                screenshotsPath.Split('\\');
                            var g = string.Empty;

                            for (var i = 0;
                                i < pathG.Length - 2;
                                i++) g += pathG[i] + "\\";

                            screenshotsPath = g;
                            var name1 = applicationName.Trim();


                            var relativePath = screenshotsPath +
                                "screenshots\\" + name1 + "\\";
                            ScreenshotName =
                                relativePath + name1 + "-" +
                                string.Format(
                                    "{0:yyyyMMddHHmmssFFF}",
                                    DateTime.UtcNow)
                                + "testnr_" + SetTestnr +
                                ".png";
                            General.LogMessage(
                                "Screenshot name: " +
                                ScreenshotName, 1);
                            driver.TakeScreenshot()
                                .SaveAsFile(ScreenshotName,
                                    ScreenshotImageFormat.Png);
                            break;

                        case "logout":
                            driver.Manage().Cookies
                                .DeleteAllCookies();
                            break;
                    }

            if (testElement == string.Empty)
                General.LogMessage("", 1, testname,
                    Convert.ToInt32(testnr), testcase,
                    testaction, testAttribute);
            else
                General.LogMessage(
                    "Element '" + testElement + "' gevonden!",
                    1, testname,
                    Convert.ToInt32(testnr), testcase,
                    testaction, testAttribute);
        }

        /*var appendtext = GetCurrentDir(2) + "\\" +
                         vestandsnaam + ".js";
        var sw = File.CreateText(appendtext);
        sw.WriteLine(inhoud);

        sw.Close(); */
    }
    catch (Exception exception)
    {
        General.LogMessage(exception.Message, 4,
            table.Rows[intx]["testname"].ToString().Trim(),
            Convert.ToInt32(table.Rows[intx]["testnr"]
                .ToString().Trim()), string.Empty,
            machinestatic);
    }
}
else
{
    MessageBox.Show("e.Message");
    break;
}
}
}


/// <summary>
///     Verifies the specified element name.
/// </summary>
/// <param name="elementName">Name of the element.</param>
/// <returns><c>true</c> if XXXX, <c>false</c> otherwise.</returns>
/// <autogeneratedoc />
/// TODO Edit XML Comment Template for verify
public static bool verify(string elementName)
{
try
{
return true;
}
catch
{
return false;
}
}
}
}
```