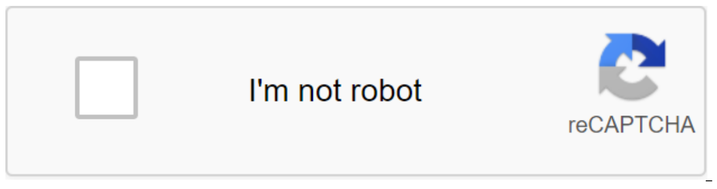
Selenium webdriver with c



Continue

Selenium 2.0 is a functional testing automation tool that includes two previously independent projects, Selenium Remote Control and WebDriver. In Selenium 2.0, you can use all the charms of both Selenium WebDriver and Selenium RC (in WebDriver compatibility mode). The developers of the tool recommend using Selenium WebDriver in places where Selenium RC can't cope, of course, unless there are other factors that can influence the choice of automation tool. When working on Selenium 2.0, the main work was on Selenium WebDriver and compatibility mode with Selenium RC. Selenium RC itself has not undergone much changes, mostly fixed old bugs, so most likely will have to rewrite some tests (remove crutches, etc.) if you are going to use RC in compatibility mode. The main difference that divides WebDriver and Selenium RC is the way you interact with the browser. Selenium RC sends commands to the browser using a special JavaScript Selenium Core kernel. This approach allows for cross-browsers (Selenium 1.0 can work with different browsers with relative ease). In this post, I described the Selenium testing tools. WebDriver, unlike Selenium RC, communicates with the browser through a native interface. Each browser has its own native interface, and this imposes certain difficulties with the support of different browsers in WebDriver. Instead, it provides a number of benefits, such as speed, user actions are emulated as accurately as possible (for example, Tests on WebDriver do not see hidden interface elements). That is, Selenium 2.0 actually combines all the pros (and disadvantages) tools listed earlier. The Selenium RC architecture can be presented as follows: Auto tests - Selenium RC Server - How Selenium RC works: Selenium WebDriver doesn't have a Selenium RC Server. But the Driver layer is added, which is responsible for the interaction with the browser. That's how Selenium WebDriver works: The main advantage of Selenium WebDriver is that it uses drivers adapted to a particular browser, meaning Selenium WebDriver works with each browser on an individual program. This improves the stability of the tests (as they are sharpened under a particular browser), tests become easier to write and maintain, increasing the speed of their work. Selenium WebDriver uses native commands (trying to fully emit user actions), which is an important advantage over Selenium RC. At the moment, there are the following drivers: FirefoxDriver; InternetExplorerDriver; ChromeDriver; OperaDriver is not yet available for use; SafariDriver - not yet available for use; HtmlUnitDriver is a cross-platform driver on java that does not require anyone to install a browser; AndriodDriver is a mobile browser. The ability to test web applications for phones is another feature in Selenium 2.0; IphoneDriver is a mobile browser. List of supported languages in Selenium WebDriver: C, Java, Ruby, Phyton. About the main commands in Selenium WebDriver I will try to write in the following notes, it is worth noting that some principles of work compared to c Selenium 1.0 have changed for the better. For example, WebDriver offers a fairly flexible way of working with expectations of events on the This is because WebDriver is primarily focused on working with complex, ajax-overloaded web interfaces. Developers offer two waiting mechanisms (Explicit Waits, Implicit Waits), here you can see the description and examples. Explicit Waits, using the WebDriverWait class, represents a suba of a virtually cyclical wait for an event. Implicit Waits - all item search teams automatically become waiting. Naturally, you can use your methods of waiting, well, or (which is very not desirable) Thread.sleep(). By the way, Selenium WebDriver knows how to work with HTML 5! Also, very pleased with the changed mechanism of work with locators - work has become even more convenient. Selenium WebDriver has added new types of locators: partionalLinkText, tagName, name. Dom locators are not supported. In this post, I painted the types of locators in Selenium 1.0. Full list of locators in Selenium 2.0: By.id - id (unique identifier) of a page element is used as a locator; By.name - the name of the page element is used as a locator; By.xpath - used to find an XPath item; By.tagName - search by the name of HTML tag; By.className - search for an item class by CSS; By.cssSelector - this type of locator is based on style table descriptions (CSS); By.linkText - search for a link with the following text; By.partionalLinkText - search for part of the link with the specified text. In order to start developing auto tests on Selenium 2.0 under .Net you need to download from here a set of libraries for .Net. In the downloaded archive should be the following libraries: Castle.Core.dll, Ionic.zip.Reduced.dll, Newtonsoft.Json.Net35.dll, Selenium.WebDriverBackedSelenium.dll, ThoughtWorks.Selenium.Core.dll, WebDriver.Support.dll. Квинси Квинси, Квинси Квинси, Квинси Зенит и Зенит. Nyanithi - zurab M. zenith and zenith. NUnit: nmock.dll, nunit.core.dll, nunit. The visual studio of zurab M. You can then start writing the first test. Kinsey Kualen, quincy quincy One, quincy quincy - Selen WebDriver; Search (Senit); Check the search results. Selen WebDriver: using the system; With System.Text with NUnit.Framework; using Open'A. Selenium; Open-A.Selenium.Firefox namespace SeleniumTests - TestFixture public class Untitled - private driver IWebDriver; Private check of StringBuilderErrs; Private string base (SetUp) Public Void SetupTest () / / You can use any WebDriver implementation. Firefox is used here as an example of a driver and the new FirefoxDriver baseURL - ; The new StringBuilder TearDown Public Void TeardownTest Catch (Exception) - / Ignore errors if you can't close your browser - Assert.AreEqual (, verificationErrors.ToString); (Test) public void TheUntitledTest () - // Open Driver ./qa. Navigation. GoToUrl (Basurl); AssertTitle Trap Catcher Assert.AreEqual type name Is Driver Selen WebDriver. FindElement (By.Name (s)). SendKeys (Selen WebDriver); Click id'search for The Driver. FindElement (By.Id (search)). Click the button AssertTitle Search Results Selen WebDriver-Error Catcher .AreEqual (Selena WebDriver-Catcher Bugs Search Results, Driver. assertText is a real Assert.IsTrue (driver. PageSource.Contains; LifeLink is a real driver. FindElement (By.LinkText) 2.0)); Now let's try to run the test. Before you do, knock down the project in Visual Studio and check that there are no bugs. If all is well, go further, if not - write in comments or on email. Start the previously installed NUnit; In NUnit, we choose File - 'gt; Open Project...; And in the dialog window we point the way to .dll of our test (... binDebugSeleniumTest.dll); Load the project and presses the Run button (it will be hard not to notice). As a result of these non-andch actions, we should start a previously written auto test on Selenium. This is what a NUnit window with a downloaded test should look like: This example used the following Selenium WebDriver commands: void GoToUrl (string url) to go to the address listed in baseURL; string Title q get; - returns Title active page; FindElement (Open.Selenium.By by) is a search for a page item according to the specified type of locator. For example, driver. FindElement (By.Name(s) or driver. FindElement (By.LinkText(The list of supported types of locators is provided above; Send voidKeys (string text) - enter the value into the text box; void Click () - Clicking on a page item public bool Contains (string value) - returns a value indicating whether the entered value is part of the object. Selenium 2.0 also created a plan to move from Selenium 1.0 to Selenium 2.0 in WebDriver mode. The plan is something like this: Use the Selenium 1.0 interface emulator for Java and .Net. To do this, you need to use the initialization of WebDriverBackedSelenium instead of DefaultSelenium. Judging by the comments, there may be difficulties at this stage (and are likely to arise). Problems may arise due to the fact that the commands in Selenium 2.0 are slightly different (fixes of old problems, plus theoretically new problems) from Selenium 1.0. after this stage, the stabilization phase is likely to follow. Zenith - zenith 2.0. zindin zidan and zelenium, Villages, WebDriverBackedSelenium, WebDriver API:WebDriver driver (((WrapsDriver) selenium).getWrappedDriver(); Zenith - zenith - 1.0, Selen - 2.0. Kinsey Kinsey Kash (Senit, Selenium Server. - Selenium Server. - Selen 1.0, Seleniy 2.0 (web driver) (10 x 10 m/ using the system; via System.Text; via NUnit.Framework; using Open'A. Selenium; Open-A.Selenium.Firefox Using selenium; namespace SeleniumTests - TestFixture public class Untitled - private driver IWebDriver; Private check of StringBuilderErrs; Private string base ISelenium private selenium; (SetUp) Public Void SetupTest () / / You can use any WebDriver implementation. Firefox is used here as an example of a driver and the new FirefoxDriver The basic URL used by selenium to address the relative URLs of baseURL is ; Creating selenium for Selenium implementation - the new WebDriverBackedSelenium (driver, baseURL); Selenium. Beginning The new StringBuilder TearDown Public Void TeardownTest () Stop Catch (Exception) - / Ignore errors if you can't close your browser - Assert.AreEqual (, verificationErrors.ToString); (Test) public void TheUntitledTest () - // Open Driver ./qa. Navigation. GoToUrl (Basurl); AssertTitle Trap Catcher Assert.AreEqual GetTitle ()); type name Is Driver Selen WebDriver. FindElement (By.Name (s)). SendKeys (Selen WebDriver); Click id'search for Selen'ssubmit. Click (search); Selenium. WaitForPageToLoad (3000); AssertTitle Search Results Selen WebDriver-Error Catcher .AreEqual (Selena WebDriver-Catcher Bugs Search Results, Driver. Text is now Search)); LifeLink is a real driver. FindElement (By.LinkText (Selen 2.0); - 2.0 euros: / You can use any WebDriver implementation. Firefox is used here as an example of a WebDriver driver - the new FirefoxDriver The basic URL used by selenium to address string baseUrl relative URLs is ; Creating Selena Selena's implementation - the new WebDriverBackedSelenium (driver, baseUrl); Perform with selenium.open ( ); selenium.type (name'q, cheese); selenium.click Return the basic implementation of WebDriver. This will apply to the same instance of WebDriver as the variable driver above. The driver of WebDriverInstance ((WebDriverBackedSelenium) selenium.getUnderlyingWebDriver (); Finally, close the browser.