InternetExplorerDriver

jimevans edited this page on Aug 27, 2015 · 3 revisions

Internet Explorer Driver

The InternetExplorerDriver is a standalone server which implements WebDriver's wire protocol. This driver has been tested with IE 7, 8, 9, 10, and 11 on appropriate combinations of Vista, Windows 7, Windows 8, and Windows 8.1. As of 15 April 2014, IE 6 is no longer supported.

The driver supports running 32-bit and 64-bit versions of the browser. The choice of how to determine which "bit-ness" to use in launching the browser depends on which version of the IEDriverServer.exe is launched. If the 32-bit version of IEDriverServer.exe is launched, the 32-bit version of IE will be launched. Similarly, if the 64-bit version of IEDriverServer.exe is launched, the 64-bit version of IE will be launched.

Installing

You do not need to run an installer before using the InternetExplorerDriver, though some configuration is required. The standalone server executable must be downloaded from the Downloads page and placed in your PATH.

Pros

- Runs in a real browser and supports Javascript

Cons

- Obviously the InternetExplorerDriver will only work on Windows!
- Comparatively slow (though still pretty snappy :)

Command-Line Switches

As a standalone executable, the behavior of the IE driver can be modified through various command-line arguments. To set the value of these command-line arguments, you should consult the documentation for the language binding you are using. The command line switches supported are described in the table below. All - `<switch>` , -- `<switch>` and / `<switch>` are supported.

Switch	Meaning
--port= `<portNumber>`	Specifies the port on which the HTTP server of the IE driver will listen for commands from language bindings. Defaults to 5555.
--host= `<hostAdapterIPAddress>`	Specifies the IP address of the host adapter on which the HTTP server of the IE driver will listen for commands from language bindings. Defaults to 127.0.0.1.

Pages 67
Find a Page...
Advanced User Interactions
Architectural Overview
Automation Atoms
Bot Style Tests
Buck
Building SeIDE
Building WebDriver
ChromeDriver
Committer Git Workflow
Continuous Integration
Continuous Integration System
Crazy Fun Build
Cygwin And WebDriver
Design Patterns
DesiredCapabilities
Show 52 more pages...

Switch	Meaning
--log-level= `<logLevel>`	Specifies the level at which logging messages are output. Valid values are FATAL, ERROR, WARN, INFO, DEBUG, and TRACE. Defaults to FATAL.
--log-file= `<logFile>`	Specifies the full path and file name of the log file. Defaults to stdout.
--extract-path= `<path>`	Specifies the full path to the directory used to extract supporting files used by the server. Defaults to the TEMP directory if not specified.
--silent	Suppresses diagnostic output when the server is started.

Important System Properties

The following system properties (read using `System.getProperty()` and set using `System.setProperty()` in Java code or the " `-DpropertyName=value` " command line flag) are used by the InternetExplorerDriver:

Property	What it means
webdriver.ie.driver	The location of the IE driver binary.
webdriver.ie.driver.host	Specifies the IP address of the host adapter on which the IE driver will listen.
webdriver.ie.driver.loglevel	Specifies the level at which logging messages are output. Valid values are FATAL, ERROR, WARN, INFO, DEBUG, and TRACE. Defaults to FATAL.
webdriver.ie.driver.logfile	Specifies the full path and file name of the log file.
webdriver.ie.driver.silent	Suppresses diagnostic output when the IE driver is started.
webdriver.ie.driver.extractpath	Specifies the full path to the directory used to extract supporting files used by the server. Defaults to the TEMP directory if not specified.

Required Configuration

- The IEDriverServer exectuable must be downloaded and placed in your PATH.
- On IE 7 or higher on Windows Vista or Windows 7, you must set the Protected Mode settings for each zone to be the same value. The value can be on or off, as long as it is the same for every zone. To set the Protected Mode settings, choose "Internet Options..." from the Tools menu, and click on the Security tab. For each zone, there will be a check box at the bottom of the tab labeled "Enable Protected Mode".
- Additionally, "Enhanced Protected Mode" must be disabled for IE 10 and higher. This option is found in the Advanced tab of the Internet Options dialog.
- The browser zoom level must be set to 100% so that the native mouse events can be set to the correct coordinates.
- For IE 11 *only*, you will need to set a registry entry on the target computer so that the driver can maintain a connection to the instance of Internet Explorer it creates. For 32-bit Windows installations, the key you must examine in the registry editor is `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\Main\FeatureControl\FEATURE_BFCACHE` . For 64-bit Windows installations, the key is `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Internet`

`Explorer\Main\FeatureControl\FEATURE_BFCACHE`. Please note that the `FEATURE_BFCACHE` subkey may or may not be present, and should be created if it is not present. **Important:** Inside this key, create a DWORD value named `iexplore.exe` with the value of 0.

Native Events and Internet Explorer

As the InternetExplorerDriver is Windows-only, it attempts to use so-called "native", or OS-level events to perform mouse and keyboard operations in the browser. This is in contrast to using simulated JavaScript events for the same operations. The advantage of using native events is that it does not rely on the JavaScript sandbox, and it ensures proper JavaScript event propagation within the browser. However, there are currently some issues with mouse events when the IE browser window does not have focus, and when attempting to hover over elements.

Browser Focus

The challenge is that IE itself appears to not fully respect the Windows messages we send the IE browser window (WM_MOUSEDOWN and WM_MOUSEUP) if the window doesn't have the focus. Specifically, the element being clicked on will receive a focus window around it, but the click will not be processed by the element. Arguably, we shouldn't be sending messages at all; rather, we should be using the SendInput() API, but that API explicitly requires the window to have the focus. We have two conflicting goals with the WebDriver project.

First, we strive to emulate the user as closely as possible. This means using native events rather than simulating the events using JavaScript.

Second, we want to not require focus of the browser window being automated. This means that just forcing the browser window to the foreground is suboptimal.

An additional consideration is the possibility of multiple IE instances running under multiple WebDriver instances, which means any such "bring the window to the foreground" solution will have to be wrapped in some sort of synchronizing construct (mutex?) within the IE driver's C++ code. Even so, this code will still be subject to race conditions, if, for example, the user brings another window to the foreground between the driver bringing IE to the foreground and executing the native event.

The discussion around the requirements of the driver and how to prioritize these two conflicting goals is ongoing. The current prevailing wisdom is to prioritize the former over the latter, and document that your machine will be unavailable for other tasks when using the IE driver. However, that decision is far from finalized, and the code to implement it is likely to be rather complicated.

Hovering Over Elements

When you attempt to hover over elements, and your physical mouse cursor is within the boundaries of the IE browser window, the hover will not work. More specifically, the hover will appear to work for a fraction of a second, and then the element will revert back to its previous state. The prevailing theory why this occurs is that IE is doing hit-testing of some sort during its event loop, which causes it to respond to the physical mouse position when the physical cursor is within the window bounds. The WebDriver development team has been unable to discover a workaround for this behavior of IE.

Clicking `<option>` Elements or Submitting Forms and alert()

There are two places where the IE driver does not interact with elements using native events. This is in clicking `<option>` elements within a `<select>` element. Under normal circumstances, the IE driver calculates where to click based on the position and size of the element, typically as returned by the JavaScript getBoundingClientRect() method. However, for `<option>` elements, getBoundingClientRect() returns a rectangle with zero position and zero size. The IE driver handles this one scenario by using the click() Automation Atom, which essentially sets the .selected

property of the element and simulates the onChange event in JavaScript. However, this means that if the onChange event of the `<select>` element contains JavaScript code that calls alert(), confirm() or prompt(), calling WebElement's click() method will hang until the modal dialog is manually dismissed. There is no known workaround for this behavior using only WebDriver code.

Similarly, there are some scenarios when submitting an HTML form via WebElement's submit() method may have the same effect. This can happen if the driver calls the JavaScript submit() function on the form, and there is an onSubmit event handler that calls the JavaScript alert(), confirm(), or prompt() functions.

This restriction is filed as issue 3508 (on Google Code).

Multiple instances of InternetExplorerDriver

With the creation of the IEDriverServer.exe, it should be possible to create and use multiple simultaneous instances of the InternetExplorerDriver. However, this functionality is largely untested, and there may be issues with cookies, window focus, and the like. If you attempt to use multiple instances of the IE driver, and run into such issues, consider using the RemoteWebDriver and virtual machines.

There are 2 solutions for problem with cookies (and another session items) shared between multiple instances of InternetExplorer.

The first is to start your InternetExplorer in private mode. After that InternetExplorer will be started with clean session data and will not save changed session data at quiting. To do so you need to pass 2 specific capabilities to driver: `ie.forceCreateProcessApi` with `true` value and `ie.browserCommandLineSwitches` with `-private` value. Be note that it will work only for InternetExplorer 8 and newer, and Windows Registry `HKLM_CURRENT_USER\\Software\\Microsoft\\Internet Explorer\\Main` path should contain key `TabProcGrowth` with `0` value.

The second is to clean session during InternetExplorer starting. For this you need to pass specific `ie.ensureCleanSession` capability with `true` value to driver. This clears the cache for all running instances of InternetExplorer, including those started manually.

Running IEDriverServer.exe Remotely

The HTTP server started by the IEDriverServer.exe sets an access control list to only accept connections from the local machine, and disallows incoming connections from remote machines. At present, this cannot be changed without modifying the source code to the IEDriverServer.exe. To run the Internet Explorer driver on a remote machine, use the Java standalone remote server in connection with your language binding's equivalent of RemoteWebDriver.

Running IEDriverServer.exe Under a Windows Service

Attempting to use IEDriverServer.exe as part of a Windows Service application is expressly unsupported. Service processes, and processes spawned by them, have much different requirements than those executing in a regular user context. IEDriverServer.exe is explicitly untested in that environment, and includes Windows API calls that are documented to be prohibited to be used in service processes. While it may be possible to get the IE driver to work while running under a service process, users encountering problems in that environment will need to seek out their own solutions.