**Chrome Driver**

Developed in collaboration with the Chromium team, the ChromeDriver is a standalone server which implements WebDriver's [wire protocol](http://code.google.com/p/selenium/wiki/JsonWireProtocol).

**Requirements**

The ChromeDriver controls the browser using Chrome's automation proxy framework. Consequently, the ChromeDriver is only compatible with Chrome version 12.0.712.0 or newer.

The server expects you to have Chrome installed in the default location for each system:

|  |  |
| --- | --- |
| **OS** | **Expected Location of Chrome** |
| Linux | /usr/bin/google-chrome1 |
| Mac | /Applications/Google\ Chrome.app/Contents/MacOS/Google\ Chrome |
| Windows XP | %HOMEPATH%\Local Settings\Application Data\Google\Chrome\Application\chrome.exe |
| Windows Vista | C:\Users\%USERNAME%\AppData\Local\Google\Chrome\Application\chrome.exe |

1 For Linux systems, the ChromeDriver expects /usr/bin/google-chrome to be a symlink to the actual binary. See also the section on [overriding the Chrome binary location](http://code.google.com/p/selenium/wiki/ChromeDriver#Overriding_the_Chrome_binary_location) .

**Getting Started**

To get set up, first [download](http://code.google.com/p/selenium/downloads/list) the appropriate prebuilt server. Make sure the downloaded server can be located on your PATH or specify its location via the webdriver.chrome.driver system property. Finally, all you need to do is create a new ChromeDriver instance:

WebDriver driver = new ChromeDriver();  
driver.get("http://www.google.com");

**Running the server in a child process**

You may notice that the ChromeDriver class is merely a convenience class that starts the server upon creation and shuts it down when you call[quit](http://selenium.googlecode.com/svn/trunk/docs/api/java/org/openqa/selenium/WebDriver.html#quit%28%29). While the server is light weight, starting and stopping it multiple times will add a noticeable delay to a larger test suite. To compensate for this, you can directly control the life and death of the server using the ChromeDriverService:

import static org.junit.Assert.assertEquals;  
  
import org.junit.After;  
import org.junit.AfterClass;  
import org.junit.Before;  
import org.junit.BeforeClass;  
import org.junit.runner.RunWith;  
import org.junit.runners.BlockJUnit4ClassRunner  
import org.openqa.selenium.chrome.ChromeDriverService;  
import org.openqa.selenium.remote.DesiredCapabilities;  
import org.openqa.selenium.remote.RemoteWebDriver;  
  
@RunWith(BlockJUnit4ClassRunner.class)}  
public class ChromeTest extends TestCase {  
  
  private static ChromeDriverService service;  
  private WebDriver driver;  
  
  @BeforeClass  
  public static void createAndStartService() {  
    service = new ChromeDriverService.Builder()  
        .usingChromeDriverExecutable(new File("path/to/my/chromedriver"))  
        .usingAnyFreePort()  
        .build();  
    service.start();  
  }  
  
  @AfterClass  
  public static void createAndStopService() {  
    service.stop();  
  }  
  
  @Before  
  public void createDriver() {  
    driver = new RemoteWebDriver(service.getUrl(),  
        DesiredCapabilities.chrome());  
  }  
  
  @After  
  public void quitDriver() {  
    driver.quit();  
  }  
  
  @Test  
  public void testGoogleSearch() {  
    driver.get("http://www.google.com");  
    WebElement searchBox = driver.findElement(By.name("q"));  
    searchBox.sendKeys("webdriver");  
    searchBox.quit();  
    assertEquals("webdriver - Google Search", driver.getTitle());  
  }  
}

**Running the server as a standalone process**

Since the ChromeDriver implements the wire protocol, it is fully compatible with any [RemoteWebDriver](http://code.google.com/p/selenium/wiki/RemoteWebDriver) client. Simply start up your server, create a client, and away you go:

WebDriver driver = new RemoteWebDriver("http://localhost:9515", DesiredCapabilities.chrome());  
driver.get("http://www.google.com");