1. How to set default download place in firefox?
2. How to enable auto-download in firefox?
3. How to set default download place in Internet Explorer?
4. How to enable auto-download in Internet Explorer?
5. How to set default download place in Chrome?
6. HTTP

**How to set default download place in firefox**

FirefoxProfile f = new FirefoxProfile();

f.setPreference("browser.download.folderList", 2);

f.setPreference("browser.download.dir", "d:\\");

WebDriver driver = new FirefoxDriver(f);

**How to enable auto-download in firefox**

FirefoxProfile f = new FirefoxProfile();

f.setPreference("browser.helperApps.alwaysAsk.force", false);

f.setPreference("browser.download.manager.showWhenStarting", false);

f.setPreference("browser.helperApps.neverAsk.saveToDisk", "application/zip, application/x-zip, application/x-zip-compressed”);

WebDriver driver = new FirefoxDriver(f);

**How to set default download place in Internet Explorer**

WindowsUtils.writeStringRegistryValue("HKEY\_CURRENT\_USER\\Software\\Microsoft\\Internet Explorer\\Download Directory", "d:\\");

WindowsUtils.writeStringRegistryValue("HKEY\_CURRENT\_USER\\Software\\Microsoft\\Internet Explorer\\Main\\Default Download Directory", "d:\\");

**How to enable auto-download in Internet Explorer**

**How to set default download place in Chrome**

Map<String, Object> prefs = new HashMap<String, Object>();

prefs.put("download.default\_directory", "d:\\");

ChromeOptions options = new ChromeOptions();

options.setExperimentalOption("prefs", prefs);

DesiredCapabilities capabilities = DesiredCapabilities.chrome();

capabilities.setCapability(ChromeOptions.CAPABILITY, options);

WebDriver driver = new ChromeDriver(capabilities);

**HTTP**

Ultimately we shouldn't care if a file was downloaded or not. Instead, we should care that a file *can* be downloaded. And we can do that by using an HTTP request alongside Selenium.

With an HTTP library we can perform a header (HEAD) request for the file. Instead of downloading the file we'll receive the header information for the file which contains things like the content type and length. With this information we can easily confirm the file is what we expect without onerous configuration, local disk usage, or lengthy download times.

Let's dig with an example.

It's just a simple matter of visiting the page with download links, grabbing a URL from one of them, and performing a HEAD request with it.

    @Test

**public** **void** **downloadFileRevisitedTest()** **throws** Exception **{**

        driver**.**get**(**"<http://the-internet.herokuapp.com/download>"**);**

        String link **=** driver**.**findElement**(**By**.**cssSelector**(**".example a:nth-of-type(1)"**)).**getAttribute**(**"href"**);**

        HttpClient httpClient **=** HttpClientBuilder**.**create**().**build**();**

        HttpHead request **=** **new** HttpHead**(**link**);**

        HttpResponse response **=** httpClient**.**execute**(**request**);**

        String contentType **=** response**.**getFirstHeader**(**"Content-Type"**).**getValue**();**

**int** contentLength **=** Integer**.**parseInt**(**response**.**getFirstHeader**(**"Content-Length"**).**getValue**());**

        assertThat**(**contentType**,** is**(**"application/octet-stream"**));**

        assertThat**(**contentLength**,** is**(**not**(**0**)));**

**}**

**}**

Once we receive the response we check it's header for the content type (e.g., response.getFirstHeader("Content-Type")) and content length (e.g., response.getFirstHeader("Content-Length")) to make sure the file is the correct type and not empty.