Boni Garcia

<https://github.com/bonigarcia/webdrivermanager>

Boni

<dependency>

<groupId>io.github.bonigarcia</groupId>

<artifactId>webdrivermanager</artifactId>

<version>4.1.0</version>

<scope>test</scope>

</dependency>

Selenium

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.141.59</version>

</dependency>

NEW

WebDriverManager.chromedriver().setup();

WebDriverManager.firefoxdriver().setup();

WebDriverManager.edgedriver().setup();

WebDriverManager.operadriver().setup();

WebDriverManager.phantomjs().setup();

WebDriverManager.iedriver().setup();

WebDriverManager.chromiumdriver().setup();

NO MORE

System.setProperty("webdriver.chrome.driver", "/path/to/chromedriver");

System.setProperty("webdriver.gecko.driver", "/path/to/geckodriver");

System.setProperty("webdriver.edge.driver", "/path/to/msedgedriver.exe");

System.setProperty("webdriver.opera.driver", "/path/to/operadriver");

System.setProperty("phantomjs.binary.path", "/path/to/phantomjs");

System.setProperty("webdriver.ie.driver", "C:/path/to/IEDriverServer.exe");

| **Method** | **Description** | **Equivalent configuration key** |
| --- | --- | --- |
| driverVersion(String) | By default, WebDriverManager tries to download the proper version of a given driver binary. A concrete version can be specified using this method. | wdm.chromeDriverVersion, wdm.operaDriverVersion, wdm.internetExplorerDriverVersion, wdm.edgeDriverVersion, wdm.phantomjsDriverVersion, wdm.geckoDriverVersion, wdm.chromiumDriverVersion |
| browserVersion(String) | Alternatively to the driver version, the major version of the browser can be specified using this method. | wdm.chromeVersion, wdm.operaVersion, wdm.edgeVersion, wdm.firefoxVersion, wdm.chromiumVersion |
| cachePath(String) | Folder in which WebDriver binaries are stored (WedDriverManager cache). | wdm.cachePath |
| forceDownload() | By default, WebDriverManager finds out the latest version of the binary, and then it uses the cached version if exists. This option forces to download again the binary even if it has been previously cached. | wdm.forceDownload=true |
| useBetaVersions() | By default, WebDriverManager skip beta versions. With this method, WebDriverManager will download also beta versions. | wdm.useBetaVersions=true |
| architecture(Architecture) | By default, WebDriverManager would try to use the proper binary for the platform running the test case (i.e. 32-bit or 64-bit). This behavior can be changed by forcing a given architecture: 32-bits (Architecture.x32) or 64-bits (Architecture.x64); | wdm.architecture |
| arch32() | Force to use the 32-bit version of a given driver binary. | wdm.architecture=32 |
| arch64() | Force to use the 64-bit version of a given driver binary. | wdm.architecture=64 |
| operatingSystem(OperatingSystem) | By default, WebDriverManager downloads the binary for the same operating systems than the machine running the test. This can be changed using this method (accepted values: WIN, LINUX, MAC). | wdm.os=WIN, wdm.os=LINUX, wdm.os=MAC |
| driverRepositoryUrl(URL) | This method allows to change the repository URL in which the binaries are hosted (see next section for default values). | wdm.chromeDriverUrl, wdm.operaDriverUrl, wdm.internetExplorerDriverUrl, wdm.edgeDriverUrl, wdm.phantomjsDriverUrl, wdm.geckoDriverUrl |
| useMirror() | The [npm.taobao.org](http://npm.taobao.org/mirrors/https:/docs.travis-ci.com/user/sonarcloud/) site is a mirror which hosts different software assets. Among them, it hosts chromedriver, geckodriver, operadriver, and phantomjs driver. Therefore, this method can be used for Chrome, Firefox, Opera, and PhantomJS to force to use the taobao.org mirror. | wdm.useMirror=true |
| proxy(String) | Use a HTTP proxy for the Internet connection using the following notation: my.http.proxy:1234 or username:password@my.http.proxy:1234. This can be also configured using the environment variable environment variable HTTPS\_PROXY. | wdm.proxy |
| proxyUser(String) | Specify a username for HTTP proxy. This can be also configured using the environment variable environment variable HTTPS\_PROXY\_USER. | wdm.proxyUser |
| proxyPass(String) | Specify a password for HTTP proxy. This can be also configured using the environment variable environment variable HTTPS\_PROXY\_PASS. | wdm.proxyPass |
| ignoreVersions(String...) | Ignore some versions to be downloaded. | wdm.ignoreVersions |
| gitHubTokenName(String) | Token name for authenticated requests (see [Known issues](https://github.com/bonigarcia/webdrivermanager#known-issues)). | wdm.gitHubTokenName |
| gitHubTokenSecret(String) | Secret for authenticated requests (see [Known issues](https://github.com/bonigarcia/webdrivermanager#known-issues)). | wdm.gitHubTokenSecret |
| localRepositoryUser(String) | Specify a username for local repository. | wdm.proxyUser |
| localRepositoryPassword(String) | Specify a password for local repository. | wdm.proxyPass |
| timeout(int) | Timeout (in seconds) to connect and download binaries from online repositories | wdm.timeout |
| properties(String) | Properties file for configuration values (by default webdrivermanager.properties). | wdm.properties |
| avoidExport() | Avoid exporting JVM properties with the path of binaries (i.e. webdriver.chrome.driver, webdriver.gecko.driver, etc). Only recommended for interactive mode. | wdm.avoidExport |
| avoidOutputTree() | Avoid create tree structure for downloaded binaries (e.g. webdriver/chromedriver/linux64/2.37/ for chromedriver). Used by default in interactive mode. | wdm.avoidOutputTree |
| avoidFallback() | If some problem is detected while resolving a driver, a fallback mechanism is used by default (i.e. use the latest version from cache as driver). To method should be used to deactivate this fallback mechanism. | wdm.avoidFallback |
| avoidBrowserDetection() | Avoid checking the version of the installed browser (e.g. Chrome, Firefox) to find out the proper version of the required driver (e.g. chromedriver, geckodriver). Only recommended for WebDriverManager as Java dependency mode. | wdm.avoidBrowserDetection |
| avoidReadReleaseFromRepository() | Avoid checking the repository (e.g. [chromedriver-latest](https://chromedriver.storage.googleapis.com/LATEST_RELEASE), [msedgedriver-latest](https://msedgedriver.azureedge.net/LATEST_STABLE)) to find out the version of the required driver. | wdm.avoidReadReleaseFromRepository |
| browserPath() | As of WebDriverManager 3.0.0, versions of drivers are contained in an internal database (versions.properties) which matches the driver version to each browser version. To find the browser version, a command is executed on the shell (e.g. google-chrome --version in Linux). The path of the browser can be configured using this method | wdm.binaryPath |
| ttl(int) | As of WebDriverManager 3.1.0, resolved versions of drivers are stored in the resolution cache. These values has a expiration time based on this Time To Live (TTL) value, measured in **seconds**. By default this value is 86400 (i.e. 1 day). | wdm.ttl |
| ttlBrowsers(int) | As of WebDriverManager 4.1.0, resolved versions of browsers are stored in the resolution cache. These values has a expiration time based on this Time To Live (TTL) value, measured in **seconds**. By default this value is 3600 (i.e. 1 hour). | wdm.ttlForBrowsers |
| setVersionsPropertiesUrl(URL) | URL of the online version.properties file, used if the relationship between browser and driver is unknown in the local version of that file. By default this value targets to the master branch of GitHub. | wdm.versionsPropertiesUrl |
| useLocalVersionsPropertiesFirst() | As of WebDriverManager 3.7.1, the online version.properties file is read to check the latest relationship browser-driver. If the local version.properties want to be used instead, this method should be invoked | wdm.versionsPropertiesOnlineFirst |
| clearResolutionCache() | This methods allows to remove the resolution cache (browser and driver versions previously resolved). | wdm.clearResolutionCache |
| clearDriverCache() | This methods allows to remove the driver cache (driver previously resolved). By default, the resolution cache is located in the same folder that the driver, and therefore, it will be also cleared when calling this method. | wdm.clearDriverCache |