Bid Adieu to driver.exe and Welcome WebDriverManager API

When you started learning Selenium WebDriver, you started it with a class, main method and putting everything inside your main method. First two lines of your code inside main method were setting up your chrome driver, firefox driver or IE driver like this :

```
System.setProperty("webdriver.chrome.driver", "path to your driver location");
 WebDriver driver = new ChromeDriver();
```

Then you gained some good knowledge and you started working in real time in a framework using TestNG or Junit library without a main method. But above two lines remained as it is. There creates a problem when you have to work on multiple browsers across multiple Operating systems. You have to download binaries of all the browsers and place it inside your project. The problem magnifies more when you have to run your code on a remote machine or a CI/CD pipelines. Whenever your browser version is updated, you have to manually download the corresponding binaries for all browsers.

To simplify these challenges a library called WebDriverManager has been designed by Bonigarcia. WebDriverManager API allows you to automatically manage binaries of drivers used by your code. We will discuss usage of this library today.

Dependency:

If you are a maven user, you need to add this dependency in your pom.xml

```
<dependency>
    <groupId>io.github.bonigarcia</groupId>
    <artifactId>webdrivermanager</artifactId>
    <version>3.6.2</version>
    <scope>test</scope>
</dependency>
```

If you are a gradle user, you need to add this dependency in your gradle project:

```
dependencies {
    testCompile("io.github.bonigarcia:webdrivermanager:3.6.2")
}
```

Once you have updated your project, you can start using WebDriverManager API. Based on your browser type, you can add the following line of code to download driver of your browser.

WebDriverManager.chromedriver().setup();

WebDriverManager.firefoxdriver().setup();

WebDriverManager.iedriver().setup();

WebDriverManager.operadriver().setup();

WebDriverManager.phantomjs().setup();

WebDriverManager.edgedriver().setup();

It downloads the driver.exe in your */m2/repository/webdriver* folder. It also automatically checks your browser version and download the matching version of driver.exe.

However, if you want WebDriverManager API to download a specific version of driver.exe you can mention it like this:

WebDriverManager.chromedriver().version("2.26").setup();

And it will do the magic for you. It will download the chromedriver.exe 2.26 to your .m2 folder.

Similarly, if you want to download 32 bit version or 64 bit version of your IE driver you can mention it to the WebDriverManager and it will do the job.

WebDriverManager.iedriver().arch32().setup();

WebDriverManager.iedriver().arch64().setup();

And that's it. You have got 32 bit or 64 bit IEDriver.exe downloaded to your machine.

To give you a real time idea, a working code will look like this:

```
public class ChromeTest {

        private WebDriver driver;

        @BeforeClass
        public static void setupClass() {
            WebDriverManager.chromedriver().setup();
        }

        @Before
        public void setupTest() {
            driver = new ChromeDriver();
        }

        @After
        public void teardown() {
            if (driver != null) {
                driver.quit();
            }
        }

        @Test
        public void test() {
            // Your test code here
        }

    }
```

I would walk you through some of the commonly used WebDriverManager methods.

targetPath(String)	You can specify the folder path where you want binaries to be downloaded.
useBetaVersions()	By default, WebDriverManager don't download the beta versions. You can force WebDriverManager to download beta version using this method.
operatingSystem(OperatingSystem)	You can specify the OS for which binaries need to be downloaded if its different than the machine running the test
ignoreVersions(String...)	You can ignore few versions to be downloaded.
getVersions()	This method allows you to find out the list of all binary versions for a browser.

getBinaryPath()	This method gives you path of the latest downloaded binary.
getDownloadedVersion()	You can find the version of the latest downloaded binary.
clearCache()	You can remove all binaries downloaded by WebDriverManager.

You can find all methods and their usage here.

WebDriverManager API makes you free from worrying about the matching browser version and your driver.exe version. It automatically does the job for you. I hope now you have got a good understanding of WebDriverManager API and will start using it in your framework rather than that old System.setProperty way of setting up the driver.

That's it for now. I will be back with some other topic. Happy reading! 😊