10. Understand the core concept of browser driver classes and web driver interface

Note : Invoking a method = executing a method

Google -> selenium download -> click on official website -> you can see Java C-sharp, Ruby Python and JavaScript -> when you are using MAVEN Project, you have to definitely go maven repository dot com.

So here ( [Downloads | Selenium](https://www.selenium.dev/downloads/) ), you can download jars directly and you can import in your project.

In Real time We just take that for MAVEN repository.

If you open selenium if you click on maven information you can directly go to the MVN repository page.

There is another dependency also that is in the mvn repository TESTng ( This is one of the excellent unit testing framework for the Java.)

So there are a lot of assertions you can get it from this particular ( TestNG)library.

 TestNG supports a variety of assertions which will help us to perform strong functional testing with selenium.

Copy the TestNG dependency paste it in the pom.xml file ->save -> you can TestNG jar files in the Maven dependencies folder.

So creating classes is a very simple. ( src -> Right click -> New class )

Java classes the place where you push your selenium code

I will show you how to write selenium tests without Java class, also using test engine.

OK, so this is the Java class, just give the name of your class -> I would call it a selenium introduction and select public static void main. -> Click on Finish.

So it created one class for you with a default template of class name and static void main.

So in Java, the execution will be happening only inside this main block. {

}

So whatever code you're right inside, this only will get executed.

So the selenium code, what we are trying to write should go inside this main block to get executed,

Our first goal is invoking the browser

note that selenium runs on different browsers.

So let's say you want to start running on Chrome browser.

So to run in Chrome browser, selenium guys have released one class called Chrome Driver.

So this class have all the methods which will help you to automate in Chrome browser.

when you understand that method is present in this chrome driver class will help us to automate in Chrome browser.

what we can do is now call these methods from our code.

So from this block { }, if we are able to call the methods, like when they say methods,

 if you want

to open the browser and hit the URL, and if you want to click on anything out of hand to close the

browser and find me at any element on page, get the text somewhere in the page.

All those are different methods we have to automate.

So all the methods are nicely grabbed under this Chrome driver class to execute in Chrome browser.

So if I want to use those methods, the first thing I want to have to do is create object of that class.

A screenshot of a computer

Description automatically generated

ChromeDriver driver = new ChromeDriver ();

above is syntax to create object chrome driver is the class and you have to give them operator

Yellow : is the class

Green : you have to give new operator

because this create a memory allocation for the object what you create.

Red : give the object name. You can give driver, selenium or anything.

By default, people refer using a driver as an object name.

So once you create object of that class, using that object, we can access all the methods which are present in this chrome driver.

Why do you want to access all these methods?

Because you want to automate in Chrome browser

So we just understood that Chrome driver have all the juice inside this, so I'm creating object of that class, first to access to methods.

OK, so after I arrived, this chrome driver is something as of now, this class did not have any knowledge.

So move your cursor on this and you will see one package name called import chrome driver.

So you need to import this package so that this Java class will get knowledge about it.

Now I have a question here.

Now, Firefox also have some class like Firefox Driver

So this also gives some methods which will help us to automate in Firefox browser.

All the method names, all their functionalities will be common and the same across all the files.

If there is something called a closed method to close the browser, same close method will be there in Firefox Driver class also.

So method names should be same.

There will be no difference between the methods.

So if you write a score and if you switch this object to Firefox and this entire code will run in Firefox

So how would these classes maintain this consistency?

Now, when I say you want to write close metaphor chrome driver, that means all classes has to follow this rule.

So how every class every browser is following this contract

Exactly to make sure these methods are same.

For that, on top over these classes.

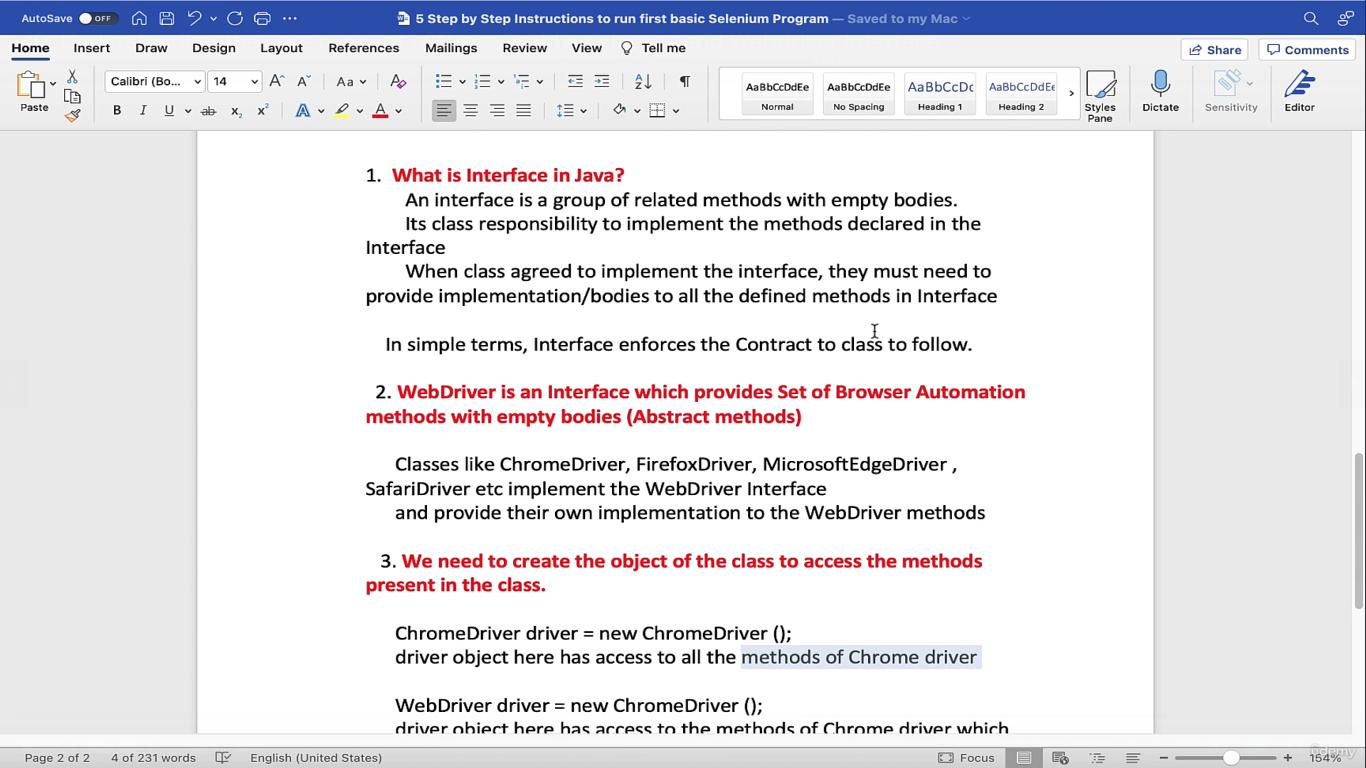
There is one interface card web driver.

OK, now what is the interface?

We just realized that there is something sitting top over these classes which is directing what to do and what rules to follow.

And these glasses are following those rules and created the methods so that everyone uses common methods

and that headmaster here is web driver and here is nothing but interface.



 interface will expose some methods  still expose methods like this close to close the browser, get to hit url on browser, but you know what?

It only told you the method names, but it don't have any implementation.

Empty body the real code to close the browser I don't have it is simply giving the method names. It's just pushing you the contract.

Let's say if Chrome driver want to implement this interface.

So there is keyword called implements in Java.

So where they just put implements and write Webdriver.

So when chrome driver done like this, so that means Chrome driver is agreeing to implement all the methods present in this Webdriver.

So when it is agreed to contract to implement the methods in Webdriver, they are forced to implement

the methods.

What Webdriver declared here when Webdriver said close, then these guys also have to create one close method and implement body for it.

interface job is to only provide a blueprint for the class or I would say contract with

just empty bodies and classes who are implementing this.

Interfaces will go and write the actual body to make it work.

So Webdriver on top defined all the method names what you should follow, and these browsers simply

are following the rules and created their own methods implementations but with same names.

Every browser came up with one class.

Google->webdriver interface->open the official doc ->

Basically, selenium Webdriver is a tool name, right?

But technically speaking, Webdriver is one interface.

And this is famous Java interview question.

You can read it all known implementing classes.

So that means all these classes agreed to implement this interface.

So this interface provides all these methods close find element, find elements, current URL, page source.

All these are selenium methods, but they don't have anybody.

These people are going and implementing this interface and giving the right implementation to this.

If you want to automate in browser, we use driver object.

So I told you that chrome driver is now implementing Webdriver methods.

But that does not mean that chrome duty is to only implement webdriver methods.

It can also have its own class methods.

So they are agreeing to implement all webdriver methods.

On top of it, they can have their own internal class methods. Also, it's up to their wish.

It's not that they have to implement only webdriver methods.

Additionally, they can have their personal class methods also.

Now by creating object for this chrome driver class, this object have capability of accessing Webdriver

implemented methods as well as personal chrome driver class methods also.

But do we really need this now?

Because our goal is to automate selenium webdriver interface methods, right?

So by having like this, what happens is if driver also access this personal class methods, this personal

class methods will not be available in Firefox and Safari, right?

These are like belonging to this chrome driver class.

let's say reset input state and let's assume that this belongs to this.

So when you run this code in Firefox, it won't run there because this method is belonging to own chrome driver but not Webdriver.

So how do you restrict your driver to focus only on these methods?

Somehow we should be giving knowledge to access only this and make this invisible for that.

If you refer your driver object to Webdriver perfect.

If you write your code like this, then this driver will have knowledge of the methods only which webdriver

implementation of chrome driver.

Okay, so move your cursor and here you see import webdriver import this

This is the famous famous interview question.

So there are two ways of declaring chrome driver object. – in the doc two sytax are there.

It's because Webdriver is interface Chrome driver is implementing and we want to refer only Webdriver implemented methods.

If I write chrome driver like this, you can explain them that time.

This code might not work in other browsers.

If this driver is accessing personal chrome driver class methods.

Guys, don't worry only for invoking(calling) the browser it have thereafter.

Everything is straightforward

11. How to run tests in Google Chrome & Importance of Chromedriver.exe file

So there is one last step you have to do before you see the browser invoking.

selenium will not have access to invoke Chrome browser or Firefox or Safari.

So there is a strict browser restriction that any browser will not allow you to access directly.

Selenium will invoke Chrome driver.

There is one chrome driver dot exe file in windows.

We say it has executable file in mac.

So this chrome driver dot EXE is a third party library which is provided by Chrome guys here.

so Selenium guys have approached Chrome browser team asking that hey, we want to automate on

your browser, but unfortunately, due to our architecture design, we cannot invoke your browser directly

from our selenium code.

And what is the solution here?

And these people have came up with one driver file and if Selenium can invoke this file,

will internally do some logic and invoke the browser for you.

First, it will be interpreted by this dot exe file and from this dot exe file only the actions will

go on to the Chrome browser.

So that means there is middleman proxy which is helping us between selenium code and actual browser.

Selenium is not directly talking to the Chrome browser here.

It is doing all the work through this chrome driver dot exe and this should be provided by Chrome team.

Right now we just realized that this is the heart because without having this in place you just can't

execute anything on Chrome browser,

So our goal is to invoke this file, but how you are invoking from Selenium script.

So there are two ways to invoke this dot exe file before executing any selenium test.

So number one is you need to write a step here to invoke the chrome driver.

So that is first way.

Second way is if you don't write any step to invoke Chrome Driver, then there is an inbuilt library

in this chrome driver class called Selenium Manager.

Okay, so the duty of this selenium manager is to check whether if there is any step written to invoke

Chrome Driver.

If there is no step defined, then this Selenium manager library do a magic for us.

What is that magic?

So it will connect to the web and it will download the appropriate chrome driver for you automatically

and it will place in the right path so that you really need not worry about anything about this chrome

driver file.

We are just asking to invoke Chrome browser, that's all.

We are not hitting any URL or we are not basically validating anything here.

So if you run this simple one liner of code, then Chrome browser just will invoke and it will close.

Right click run as Java application.

So you are running as Java test. See it opened and closed, which is awesome.

So that means you did not give any proxy chrome driver because I'm keep on stressing that this is the

file which is responsible to invoke Chrome browser.

As you have not written any step here, Selenium manager came into picture here that connected to web

got your default chrome driver for you and it installed in your appropriate path so that it can talk to browser and it can invoke chrome browser.

One of the step the other way is you manually giving the path instead of asking Selenium manager to do that work for you,

If you are manually giving, then you have to write this step system dot set property this this method.

So what is this method?

Basically, this is the method where you are setting the global level properties so that way your script

knows where.

To look for that chrome driver.

So it takes two values.

Key and value.

So key here is you need to tell webdriver dot chrome, dot driver

So basically this is the one global level property.

In this property you have to set the path of your externally downloaded dot exe file.

So basically selenium will check for that file.

So where it will check is it will check in this property.

So in this property you need to feed with the path where that file is.

So as of now, you don't have that file, right?

Let's get it from the web.

So simply type chrome driver download.

Now there are a lot of versions available here.

Version 118 117 116 So by the time you see this page, it could be version 200, who knows, right?

So basically these are the versions of your Chrome browser.

So you cannot randomly download any chrome driver like what you want.

So that should be compatible with your browser, what you are having.

So if you are having a Chrome browser with version 115, then you should download the driver which is compatible to this version only.

So this will give you the file to download.

So here you need to unzip your file, whatever you downloaded.

When you unzip, there will be chrome driver.

This is the exe file we are looking for.

Simply the path of this file you need to give your in your script. - value

 in windows you need to mandatorily provide dot exe extension

Now you can run it again.

So what is the difference between the previous way and this?

So here, the moment your script identified that you have defined your chrome driver locally, then selenium manager will not be in active state.

That will be disabled.

So it will not check for web because you already have it in your local and you are nicely giving the

path also.

So it will directly invoke the chrome driver and that chrome driver will be ultimately responsible to

invoke your browser and from there things will go on.

when you run without using this step, it has to connect to the

web and download it and it has to run.

It might take few more seconds time, which is completely okay, but for a faster execution or to make

it quicker, I have it in my local so it will directly pick that and it will run.

So you can directly invoke the Chrome browser with just like without setting anything.

Let Selenium manager take care of that.