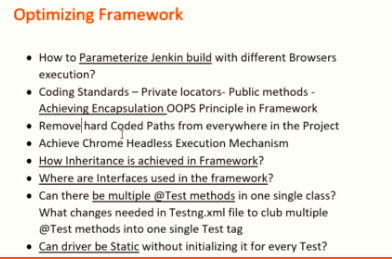
**Selenium Framework Optimization & Interview questions**

**Agenda:**

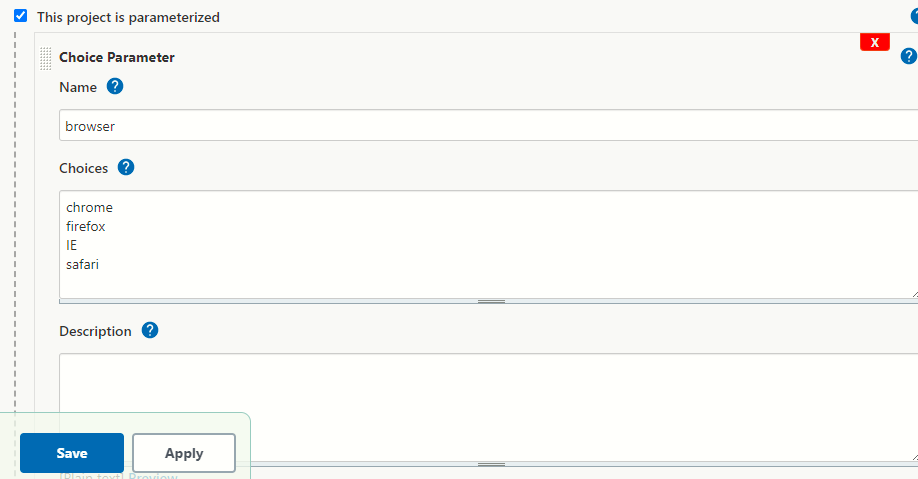


1. How to parameterize Jenkin build with Multi options profile

Mvn test -Dbrowser=chrome – this can be used in Jenkins if we want to run in particular browser

In **base** class write like String browserName = System.getProperty(“browser”); - > this takes browser from maven commands

1. Live example in parameterizing job with multiple browser options
2. Configure like below for browsers setup. Apply and Save

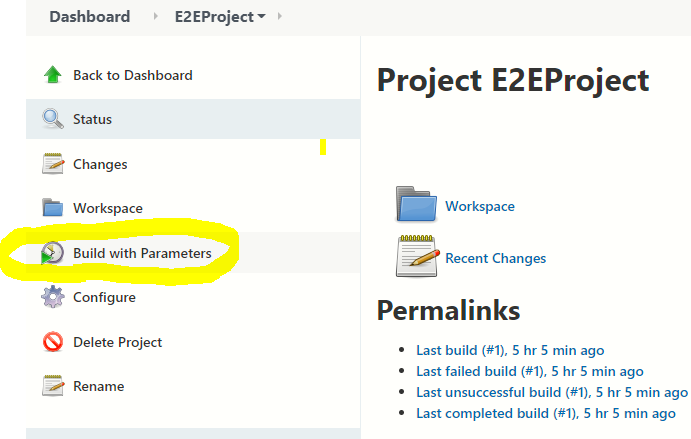


1. Give the below command and save

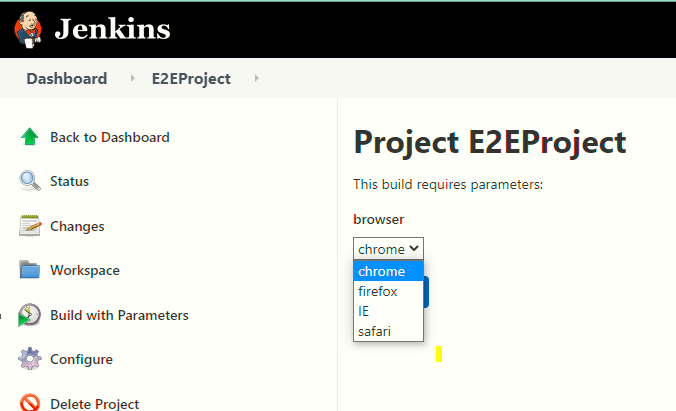
test -Dbrowser="$browser" - $ searches for Jenkins variable for browsers



1. Select the below one



1. Select the required browser and click on Build to run the project



1. Coding standards to achieve encapsulation in framework

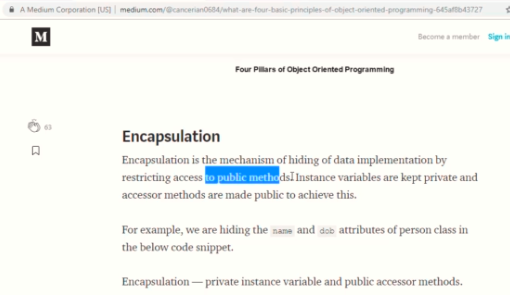
Example in framework: we are making locator variables as private and methods as public as shown below in PageObjects classes, if we make variables as public, so all the classes in framework can use directly pageobjects locators, this is not recommended and this is a Encapsulation framework. This is used to avoid dummy code in tests to not write driver.findElements code in tests classes

1. private By email=By.cssSelector("[id='user\_email']");
2. public WebElement getEmail()

{

return driver.findElement(email);

}



1. Remove hard coded paths in framework by generating paths dynamically

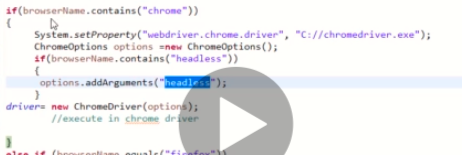
* This is to remove hard code paths in framework, this is recommended bcoz if someone pulls our code from GIT, then the script fails due to hard coded path
* eg: removing paths from ‘base class’ and set dynamic path

System.getProperty("user.dir") - > this path gives the project path

Base class – 1) FileInputStream fis=new FileInputStream(System.getProperty("user.dir")+"\\src\\main\\java\\resources\\data.properties");

2) System.*setProperty*("webdriver.chrome.driver",System.*getProperty*("user.dir") + "\\drivers\\chromedriver.exe");

1. How to execute tests in chrome head less mode?

* Do the below changes in base class
* 
* And in Jenkins add one more parameter as chromehealdess

1. Where are Inheritance, Interfaces and Statics are used in framework

* We create one ‘base’ class in resources package to write code to initialize webdriver and browsers, method for take screenshot for failed test cases, implicit and explicit waits
* We are extending ‘base’ class to the test classes by inheriting parent class – base by using ‘extends’ keyword

Eg: creating one method in test classes to call the base class methods like below

**public** **class** HomePage **extends** base {

**public** WebDriver driver;

**public** **static** Logger *log* = LogManager.*getLogger*(base.**class**.getName());

@BeforeTest

**public** **void** initialize() **throws** IOException {

driver = initializeDriver();

}

* In ‘Listeners’ class we use both extends and implements keywords – extends is used to inherit base class and implements is used for ITestListener to add unimplemented methods

**public** **class** Listeners **extends** base **implements** ITestListener

**can driver be static or non-static? – IQ**

if we create like **public** **static** WebDriver *driver,* the no need to create global variable in other classes, because if we create using static keyword then it will be applicable for all the classes, scope will be for all the classes

Eg. Global vars in other classes

**public** **class** HomePage **extends** base {

**public** WebDriver driver; - if we don’t use static in base class, then we have to create like this in all the closes to run all the tests

**Advantage –** we can save memory if we create object one time for driver by using static keyword**,** execution will be speed

**Disadvantage –** if we create like above, then we cannot do parallel testing because same class cannot access static driver reference at a time

1. Best approach on how to consolidate test cases into different classes
2. Demonstration on clubbing tests into classes in framework

* If we have 100 test cases, will you initialise base class in every class, will you execute and invoke browser 100 times?
* Will you write 100 java classes if we have 100 test cases?
* If we have 100 classes, will you write 100 test tags in testing.xml file for each class and 100 times opening and closing the browser?
* To overcome the above problems we use **page object mechanism**



* **Refer to Landing page objects class and validateTitle class for example – written multiple different tests In same class so that browser can opens only once, no need to create different classes**