

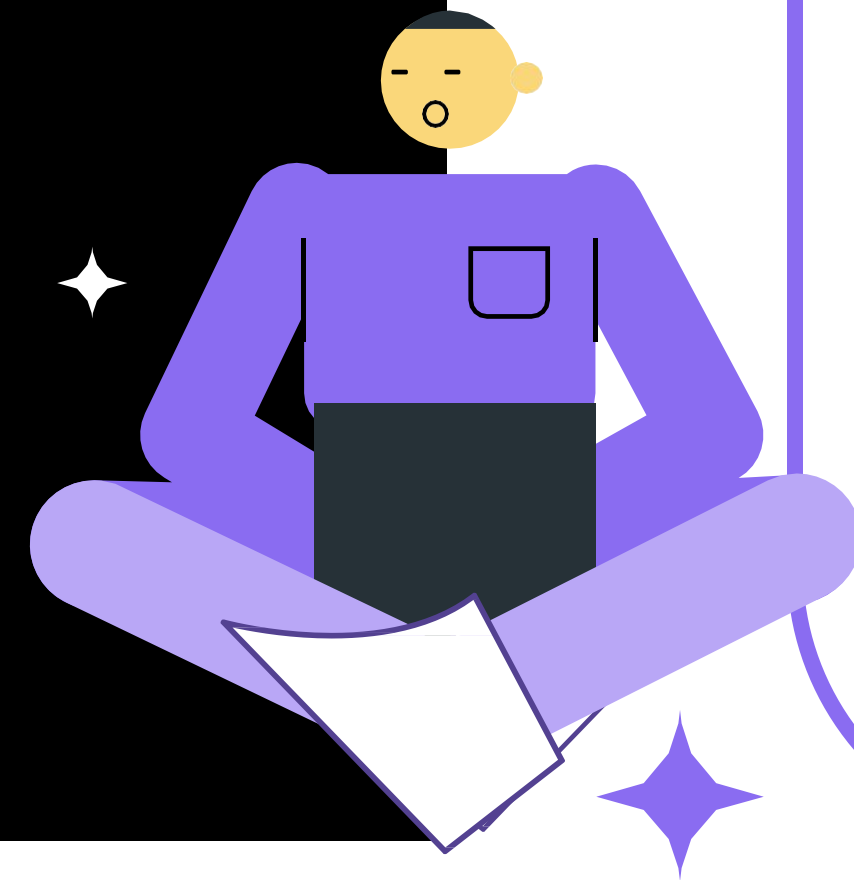
Before y After Test

Índice

1. [Before test](#)
2. [After test](#)



Vamos a conocer más sobre los métodos Before y After, aquellos métodos o rutinas que necesitan ser ejecutadas ANTES de nuestro test o DESPUÉS.



01 Before test


Before test

Para ejecutar cualquier script de prueba de Selenium, necesitarán un controlador de navegador y, para usarlo, necesitan establecer la ruta ejecutable del controlador de navegador explícitamente. Después de eso, es necesario instanciar la instancia del controlador y proceder con el caso de prueba.

```
@BeforeTest
    public void setUp() throws Exception {
        WebDriverManager.chromedriver().setup();
        driver=new ChromeDriver();
    }
```

Before test

Al principio de la clase se pueden definir variables y rellenarlas con datos:



```
class FirstTestScriptUsingSeleniumGrid {  
  
    public String username = "Digital House";  
    public String accesskey = "CTD2022";  
    public static RemoteWebDriver driver = null;  
    public String gridURL = "@https://www.digitalhouse.com/br/";  
}
```

Entidades

Las entidades son **objetos** que se mantienen en memoria durante un intervalo de tiempo y luego se persisten en la base de datos. La clase **entidad** representa **una tabla de la base de datos** y el **objeto entidad** representa una **fila de la tabla**.

Por lo general, cumple con los requisitos de la empresa para proporcionarnos la capacidad de crear los datos necesarios para realizar la prueba.

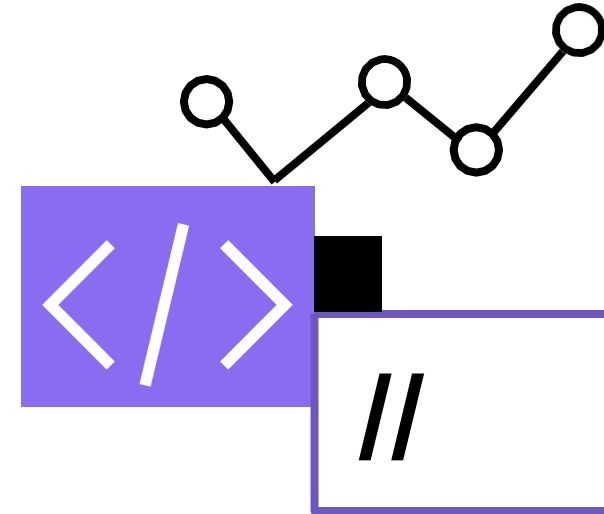
Entidades

Imaginen por ejemplo que quieren probar un sistema de cuentas bancarias, para ello pueden crear la entidad **Cuenta**, indicando los atributos y métodos necesarios.

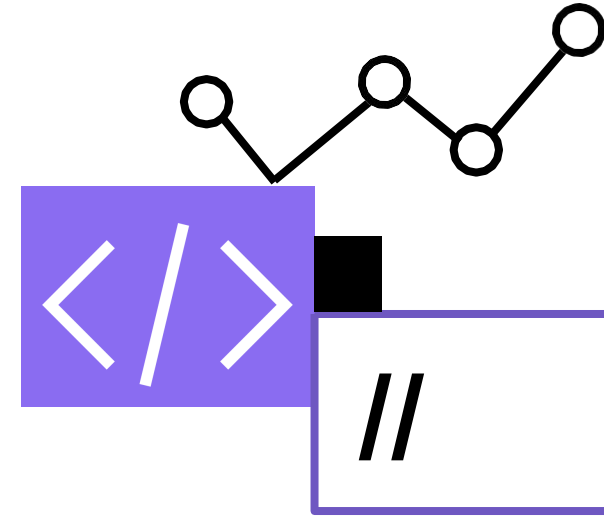
```
//Creating the bank account class

public class Account {
    private int NumberAccount;
    public int op;
    private String Name;
    private double AccountBalance;
    private double AccountLimit;

    public Account() {
        this.NumberAccount=0;
        this.Name=" ";
        this.AccountBalance=0;
        this.AccountLimit=0;
    }
    public int getNumberAccount() {
        return NumberAccount;
    }
    public void setNumberAccount(int NumberAccount) {
        this.NumberAccount = NumberAccount;
    }
    public String getName() {
        return Name;
    }
    public void setName(String Name) {
        this.Name = Name;
    }
    public double getAccountBalance() {
        return AccountBalance;
    }
    public void setAccountBalance(double AccountBalance) {
        this.AccountBalance = AccountBalance;
    }
    public double getAccountLimit() {
        return AccountLimit;
    }
    public void setAccountLimit(double AccountLimit) {
        this.AccountLimit = AccountLimit;
    }
    boolean withdrawal(double quantity) {
        if (this.AccountBalance<quantity)
            return false;
        else {
            this.AccountBalance = this.AccountBalance - quantity;
            return true;
        }
    }
    void deposit(double quantity) {
        this.AccountBalance = this.AccountBalance + quantity;
    }
    void fillinAccountData(String a, int b, float c, float l){
        this.Name = a;
        this.NumberAccount = b;
        this.AccountBalance = c;
        this.AccountLimit = l;
    }
    void showAccountInfo(){
        System.out.println("Name: " + this.getName());
        System.out.println("Number Account: " + this.getNumberAccount());
        System.out.println("Account Balance: " + this.getAccountBalance());
    }
    void showAccountBalance(){
        System.out.println("Account Balance : " + this.getAccountBalance());
    }
}
}
```



Entidades



Por ejemplo, antes de probar la posibilidad de un retiro, tendría que crear una cuenta con los datos necesarios, para saber si la cuenta tiene suficiente saldo para liberar el retiro.

```
// creating the account
Account myAccount = new Account();

// filling in the account data
myAccount.fillinAccountData ("mari", 1020, 100.50, 1000.10)
```

Entidades

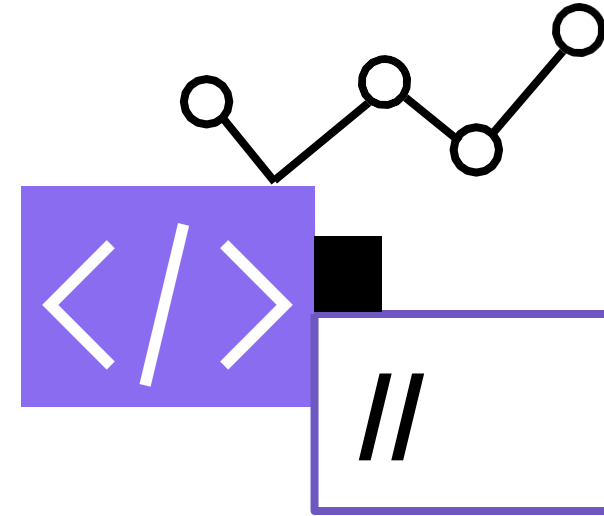
Probando el
retiro

```
@Test
public void tryWithdrawal() throws Exception {
    //given
    Account myAccount = new Account();
    myAccount.fillinAccountData ("mari", 1020, 100.50, 1000.10)

    //when

    Boolean result = myAccount.withdrawal (2000);

    //then
    assertTrue("Don't allow to withdrawal" == result);
}
```





Independientemente de la forma en que decida crear y alimentar las variables y los objetos, lo más importante es tener en cuenta qué es lo que necesitamos ejecutar antes de nuestra prueba.

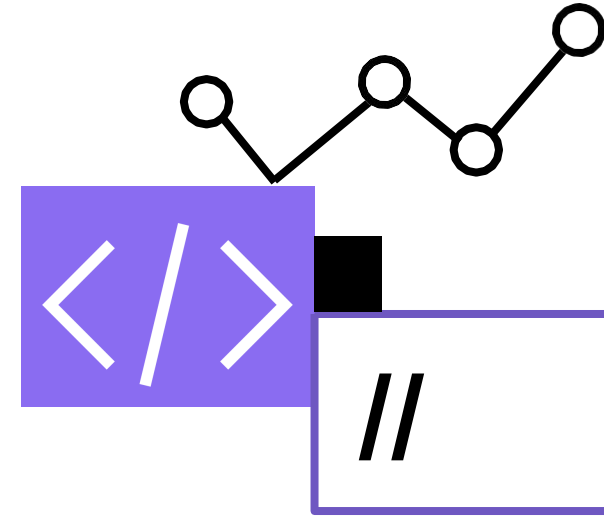
02 After test

After test

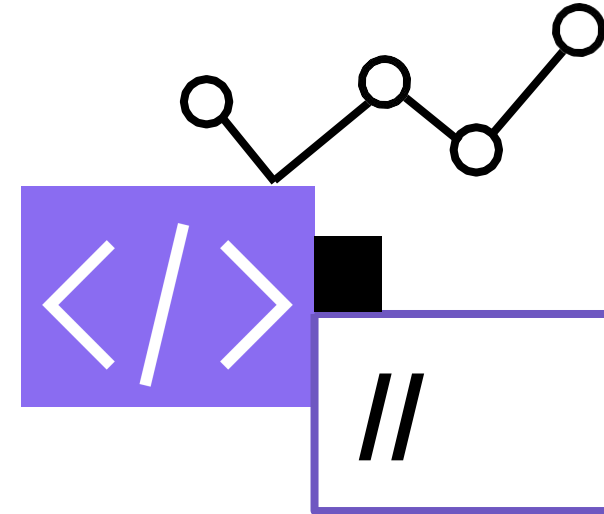
La anotación **@After** se utiliza para realizar tareas después de la ejecución de cada prueba. Por ejemplo:

- Enviar los resultados de las pruebas a un servicio de registro o supervisión.
- Cerrar el navegador.
- Crear informe después de ejecutar la prueba.
- Recoger capturas de pantalla.
- Las variables se pueden borrar o restablecer.
- Eliminar datos creados por la misma prueba.

After test



```
@AfterTest
public void closeBrowser() {
    driver.close();
    System.out.println("The driver has been closed.");
    FirstTestScriptUsingSeleniumGrid.username = "";
    FirstTestScriptUsingSeleniumGrid.accesskey = "";
}
```



Informes y registros

En caso de que una prueba falle, **es vital tener capturas de pantalla a las que referirse**. Esto ayuda a explicar el fallo al desarrollador, que puede depurarlo al instante. Del mismo modo, desde el punto de vista de los informes, para proporcionar información a las partes interesadas, **es valioso compartir con ellas los informes**, para dar visibilidad a la calidad del producto.

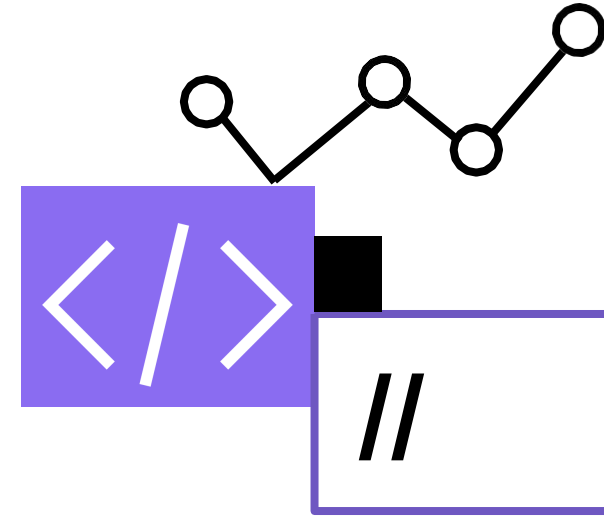
Veremos ejemplos de pruebas utilizando **Extent Report** para generar informes, registros y capturar pantallas.

Informes y registros

Intenten capturar pantallas con el siguiente código:

```
test.log(LogStatus.FAIL, test.addScreenCapture(capture(driver)) + "Test Failed");

public static String capture(WebDriver driver) throws IOException {
    File scrFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
    File Dest = new File("src/../BStackImages/" + System.currentTimeMillis()
        + ".png");
    String errflpath = Dest.getAbsolutePath();
    FileUtils.copyFile(scrFile, Dest);
    return errflpath;
}
```

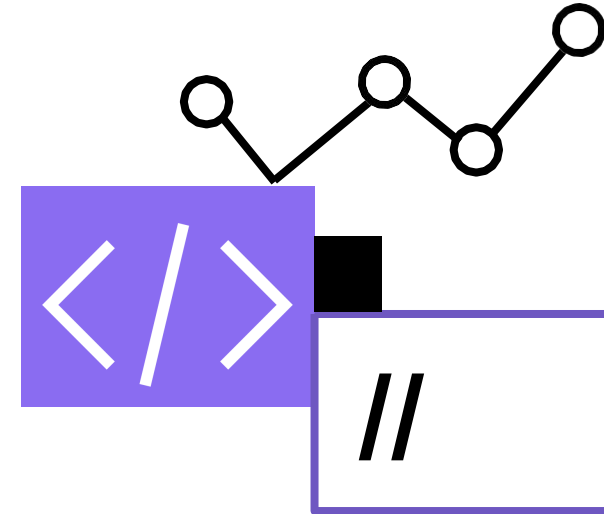


Informes y registros

getScreenshotAs(): Captura la pantalla de la instancia actual de WebDriver y la almacena en diferentes formas de salida.

```
File scrFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
```

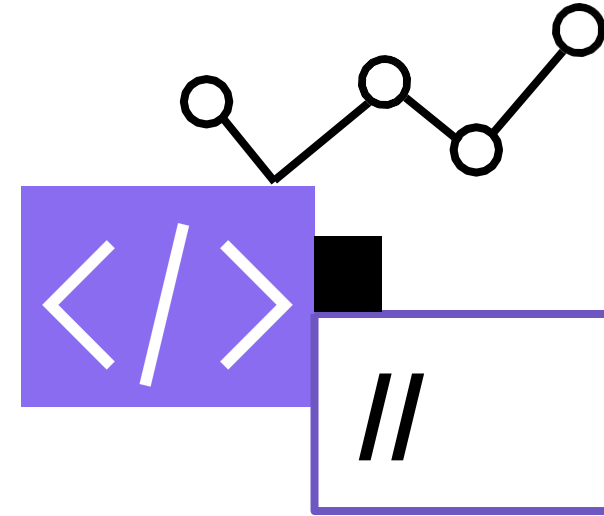
Este método devuelve un objeto de archivo que se almacenará en una variable de archivo.



Informes y registros

Esta sentencia crea una carpeta llamada '**BStackImages**' dentro de la carpeta 'src' y almacena el nombre del archivo como la hora actual del sistema.

```
File Dest = new File("src/../BStackImages/" + System.currentTimeMillis() + ".png");
```

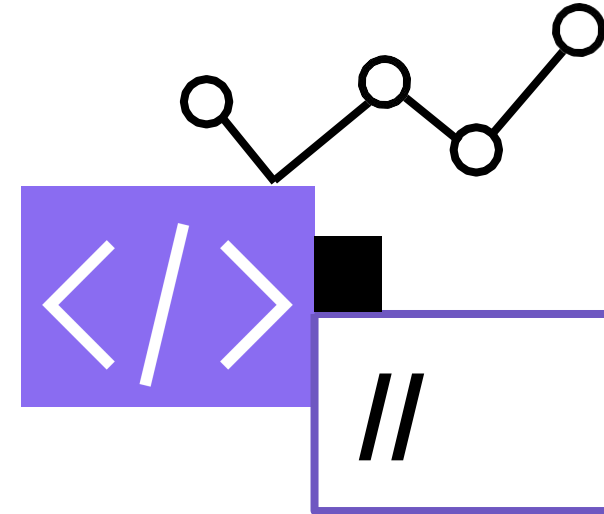


Informes y registros

Estas instrucciones copian todas las imágenes de error en la carpeta de destino.

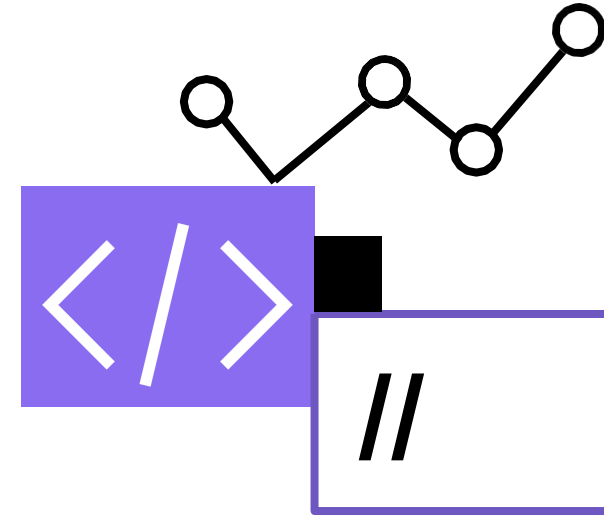


```
String errflpath = Dest.getAbsolutePath();  
FileUtils.copyFile(scrFile, Dest);  
return errflpath;
```



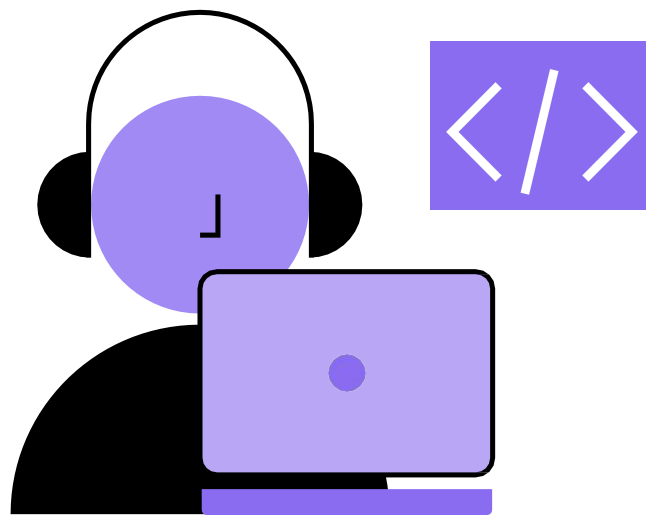
Informes y registros

Usar el método log porque utiliza el método addScreenCapture de la clase Extent Test para tomar una captura de pantalla y añadirla al Extent Report.



```
test.log(LogStatus.FAIL, test.addScreenCapture(capture(driver)) + "Test Failed");
```

Ejemplos de
script completo:



```
import org.apache.commons.io.FileUtils;
import org.openqa.selenium.By;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Reporter;
import org.junit.Assert;
import org.junit.BeforeClass;
import org.junit.Test;
import org.junit.AfterClass;

public class BrowseWikipedia {
    public static WebDriver driver;
    public Boolean shouldTheTestPass = false;

    @BeforeTest
    public static void setUp()throws Exception{
        driver = new FirefoxDriver();
    }

    @Test
    public void step01_HomePage() throws Exception{
        listEnvironmentVariables();

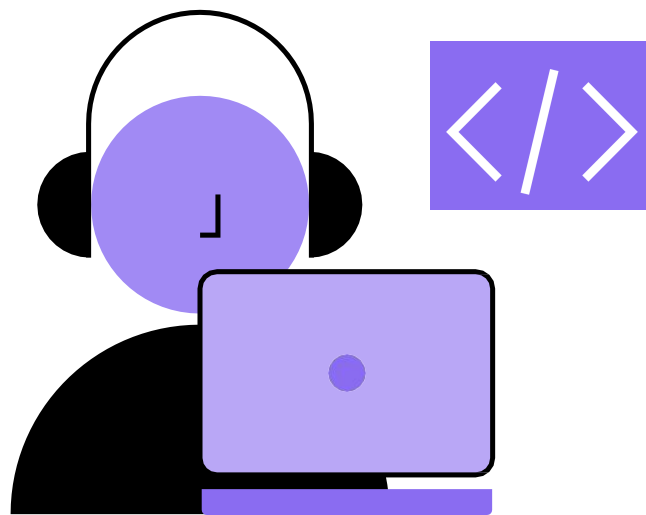
        driver.get("http://www.wikipedia.org/");
        String i = driver.getCurrentUrl();
        System.out.println(i);
        BrowseWikipedia me = new BrowseWikipedia();
        try{
            me.takeScreenShot("Step-1.png",driver);}
        catch(Exception e){
            System.out.println("Problem, "+e.toString());}
    }

    @Test
    public void step02_MainPage() throws Exception{
        driver.findElement(By.linkText("English")).click();

        BrowseWikipedia me = new BrowseWikipedia();
        try{
            me.takeScreenShot("Step-2.png",driver);}
        catch(Exception e){
            System.out.println("Problem, "+e.toString()); }
    }

    @AfterTest
    public static void tearDown()throws Exception{
        driver.quit();
    }
}
```


Ejemplos de
script completo:



```
package LambdaTest;

import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Listeners;
import org.testng.annotations.Test;

@Listeners({util.Listener.class})
class FirstTestScriptUsingWebDriver {
    public static WebDriver driver = null;
    @BeforeTest
    public void setUp() throws Exception {
        WebDriverManager.chromedriver().setup();
        driver=new ChromeDriver();
    }
    @Test
    public void firstTestCase() {
        try {
            System.out.println("Logging into Lambda Test Sign Up Page");
            driver.get("https://accounts.lambdatest.com/register");
            WebElement pageHeader= driver.findElement(By.xpath("//a[text()='Sign In']"));
            pageHeader.click();
            System.out.println("Clicked on the Sign In Button.");
        } catch (Exception e) {
        }
    }
    @AfterTest
    public void closeBrowser() {
        driver.close();
        System.out.println("The driver has been closed.");
    }
}
```

¡Muchas
gracias!