

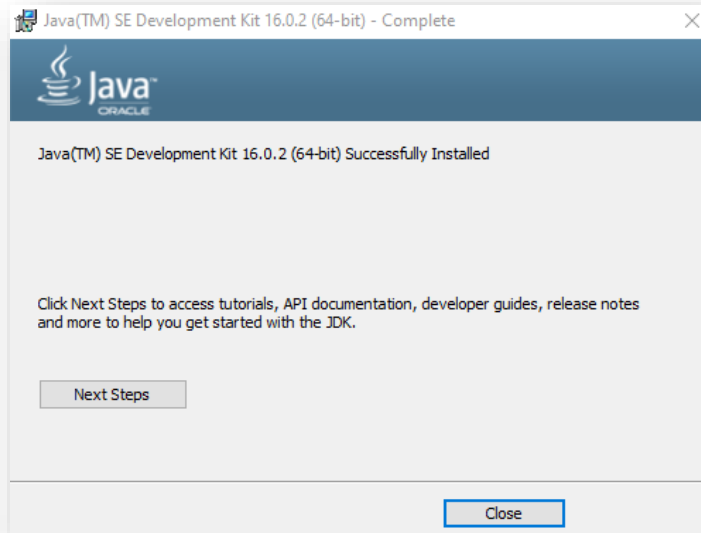
## Escenarios automatizados Selenium

Selenium es un entorno de pruebas que se utiliza para comprobar si el software que se está desarrollando funciona correctamente. Esta herramienta permite editar y depurar casos de pruebas que se pueden automatizar.

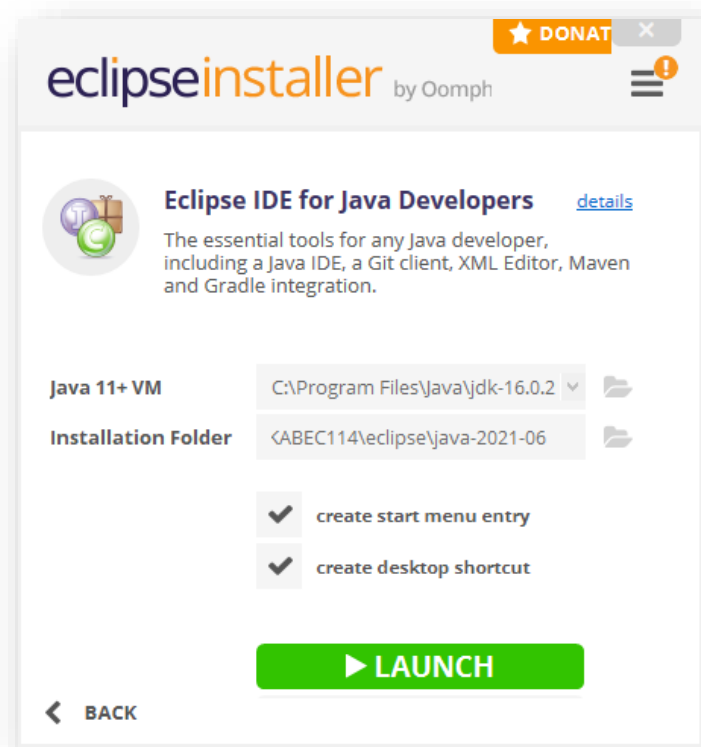
### *1.Instalación de selenium*

Para poder ejecutar selenium es necesario instalar previamente 3 componentes:

- Java JDK (software que provee herramientas de desarrollo para la creación de programas en Java)

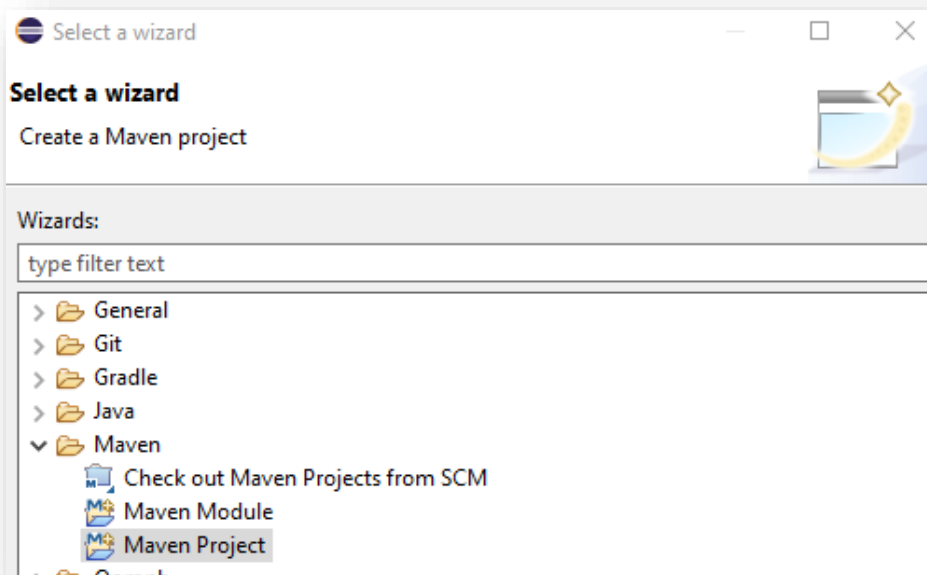


- Eclipse IDE (plataforma de desarrollo de software)



- Maven project

Se integra un proyecto de Maven para administrar la estructura del proyecto, cargar y usar las librerías del proyecto



Al configurar el proyecto agregamos la dependencia de maven y de junit al archivo pom.xml esto servirá para agregar de forma automática las librerías necesarias para ejecutar las pruebas

```
<dependencies>
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.141.59</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Creamos una nueva clase y a partir de este momento el ambiente se encuentra configurado para comenzar a correr casos de prueba con selenium

## 2. Ejemplo de Registro de usuario

Para comenzar con el *script* de pruebas es necesario importar las librerías para las funcionalidades requeridas:

`import java.io.File;`

Librería java para referirse a ficheros y directorios en el sistema de ficheros del ordenador

`import java.io.IOException;`

Librería java para que el programa pueda manejar un archivo que no existe en el momento

`import java.util.List;`

Librería para ejecutar secuencias donde se respete el orden en el que se insertan elementos, manejo de elementos duplicados o manejo de listas dinámicas

`import javax.imageio.ImageIO;`

Librería para manejo de formato de imágenes

`import org.junit.After;`

Librería para la ejecución de la superclase *After* que se ejecutara antes de todo el *script*

`import org.junit.Before;`

Librería para la ejecución de la superclase *Before* que se ejecutara después de todo el *script*

`import org.junit.Test;`

Librería para indicar que el método que la contiene es una prueba

`import org.openqa.selenium.By;`

Librería que maneja los elementos basicos de selenium

`import org.openqa.selenium.WebDriver;`

Librería que maneja la conexión y administración de un navegador web

`import org.openqa.selenium.WebElement;`

Librería que administra los elementos de una página para web para interactuar con ellos

`import org.openqa.selenium.support.ui.Select;`

Librería para manejo de elementos que necesitan ser repetidos en orden de tener un código más claro

`import org.openqa.selenium.chrome.ChromeDriver;`

Librería para manejo y ejecución de pruebas en explorador Chrome

`import ru.yandex.qatools.ashot.AShot;`

`import ru.yandex.qatools.ashot.Screenshot;`

`import ru.yandex.qatools.ashot.shooting.ShootingStrategies;`

Librerías de plugin para tomar evidencias de capturas de pantalla

Una vez indicadas todas las librerías necesarias para trabajar se declara nuestra clase e inicializamos nuestro *webdriver*

```
public class form_registro {  
    private WebDriver driver;
```

Procedemos a declarar nuestra superclase *@Before* de JUnit que contendrá los pasos iniciales previos a la ejecución de la prueba como inicializar el chromedriver para poder ejecutar las pruebas en el navegador Chrome e indicar opciones del mismo como maximizar la pantalla y dirigirse a la *url* de nuestro sitio de pruebas

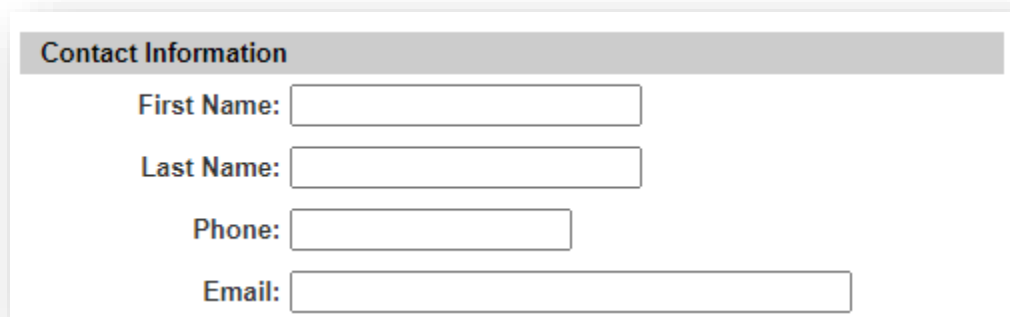
```
@Before  
public void setUp() throws Exception {  
  
    System.setProperty("webdriver.chrome.driver", "./src/test/resources/chromedriver/chromedriver.exe");  
    driver = new ChromeDriver();  
    driver.manage().window().maximize();  
    driver.get("http://demo.guru99.com/test/newtours/");  
}
```

A continuación, declaramos la superclase *@Test* donde ingresamos en cuerpo y los pasos de la prueba dentro del explorador, en este caso usamos la expresión

```
driver.findElement(By.name("Campo")).sendKeys("Valor");
```

para indicar que vamos a buscar los elementos del formulario por su propiedad *name* y posteriormente vamos a ingresar el valor en el campo a través del comando *.sendKeys* de esta forma llenamos los campos correspondientes a la sección de información personal del usuario

```
@Test  
public void test_uno() throws InterruptedException {  
  
    driver.findElement(By.LinkText("REGISTER")).click();  
  
    //información personal  
    driver.findElement(By.name("firstName")).sendKeys("Mario");  
    driver.findElement(By.name("lastName")).sendKeys("de la Rosa");  
    driver.findElement(By.name("phone")).sendKeys("5566778899");  
    driver.findElement(By.name("userName")).sendKeys("mario.delarosa@live.com.mx");  
}
```

A screenshot of a web form titled "Contact Information". It contains four input fields: "First Name:", "Last Name:", "Phone:", and "Email:". Each field is represented by a rectangular text box.

**Contact Information**

First Name:

Last Name:

Phone:

Email:

Seguimos con la información pertinente a la dirección del usuario donde usaremos los mismos comandos para llenar los campos con diferencia de una lista desplegable donde debemos indicar el país de residencia del usuario para indicar el país específico usamos el comando.


```
//dirección
driver.findElement(By.name("address1")).sendKeys("rio de los remedios 55");
driver.findElement(By.name("city")).sendKeys("puebla de juarez");
driver.findElement(By.name("state")).sendKeys("puebla");
driver.findElement(By.name("postalCode")).sendKeys("55060");
Select country = new Select (driver.findElement(By.name("country")));
country.selectByVisibleText("ANDORRA");
```

```
Select country = new Select (driver.findElement(By.name("country")));
```

Se selecciona la lista desplegable con la propiedad name=country

```
country.selectByVisibleText("ANDORRA");
```

Se selecciona el elemento que contiene el texto "ANDORRA"

A screenshot of a web form titled "Mailing Information". It contains five input fields: "Address:", "City:", "State/Province:", "Postal Code:", and "Country:". The "Country:" field is a dropdown menu with "MOZAMBIQUE" selected.

**Mailing Information**

Address:

City:

State/Province:

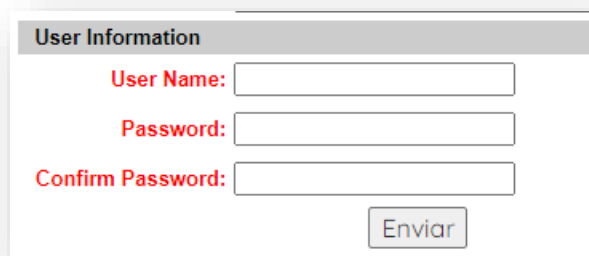
Postal Code:

Country:

Siguiendo con el flujo se integran los datos del usuario, *username*, contraseña y confirmación de contraseña usando las funciones previamente mencionadas, en este punto ingresamos una aserción para comprobar que nuestro caso de prueba se ejecute correctamente.

```
//información del usuario
driver.findElement(usernameLocator).sendKeys("mario");
driver.findElement(passwordLocator).sendKeys("contraseñ@");
driver.findElement(confirmPasswordLocator).sendKeys("contraseñ@");
driver.findElement(registerBtnLocator).click();

List<WebElement> fonts = driver.findElements(By.tagName("font"));
assertEquals("Note: Your user name is mario.", fonts.get(5).getText());
```



The image shows a web form titled "User Information". It contains three input fields: "User Name:", "Password:", and "Confirm Password:". Each field is followed by a text input box. Below the fields is a button labeled "Enviar".

Por último, se coloca la superclase *@After* donde declaramos nuestra sentencia para poder tomar capturas de pantalla del final de nuestra prueba y por ultimo cerrar el navegador.

```
@After
public void main() throws IOException {
    Screenshot Screenshot = new AShot().shootingStrategy(ShootingStrategies.viewportPasting(1000)).takeScreenshot(driver);
    ImageIO.write(Screenshot.getImage(), "jpg", new File(".\\screenshot\\fullimage.jpg"));
    driver.quit();
}
```

Antes de poder ejecutar este código es necesario colocar la dependencia dentro del archivo pom.xml para que las librerías puedan agregarse y ejecutarse dentro del proyecto.

```
<dependency>
    <groupId>ru.yandex.qatools.ashot</groupId>
    <artifactId>ashot</artifactId>
    <version>1.5.4</version>
</dependency>
```