

INTEGRACION CONTINUA

RESOLUCIÓN DE EJERCICIOS

A continuación encontrará el desarrollo de los ejercicios que resolvió en la semana. Contraste las respuestas entregadas por el docente con las desarrolladas por usted. En caso que no coincidan, y persistan dudas, le invitamos a repasar los contenidos y/ o consultar con su profesor.

Nota: para descargar los recursos necesarios para desarrollar las actividades de la semana 3 debes descargar los recursos en el siguiente enlace de Google Drive

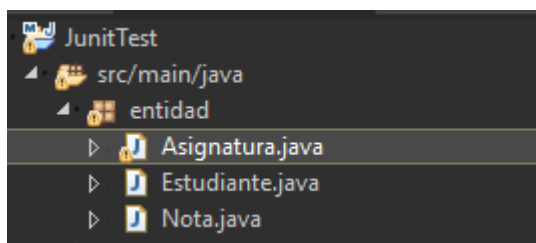
Link:

1.1 Crear un proyecto con junit y realizar pruebas unitarias 4.12 desde el repositorio de Maven a nuestro archivo POM.XML

Abrimos nuestro IDE eclipse y creamos un nuevo proyecto de Maven con nombre Junit test.

Agregamos las dependencias de Junit

Crear la siguiente estructura de clases en el directorio src/main/java



En la clase asignatura debe crear los siguientes atributos

```
private List<Nota> notas;  
private String nombre;  
private int credito;
```

creamos el constructor con validación de nombre y crédito no deben estar vacíos

```
public Asignatura(String nombre, int credito) {  
    if(nombre.trim().equals("") || nombre == null){  
        throw new IllegalArgumentException("Nombre vacio");  
    }  
    if(credito<=0){  
        throw new IllegalArgumentException("Credito debe ser mayor a 0");  
    }  
    this.nombre = nombre;  
    this.credito = credito;  
    this.notas = new ArrayList<Nota>();  
}
```

Escribimos los métodos get de nuestra clase

```
public List<Nota> getNotas() {  
    return notas;  
}  
public String getNombre() {  
    return nombre;  
}  
public int getCredito() {  
    return credito;  
}
```

Escribimos los métodos de tipo double promedio y avance.

```
public double promedio(){  
    double suma=0;  
    double ponderacionAcum = 0;  
    for (Nota nota : notas) {  
        suma = suma + ((Nota) nota).getValor()*nota.getPonderacion();  
        ponderacionAcum = ponderacionAcum + nota.getPonderacion();  
    }  
    return suma/ponderacionAcum;  
}  
public double avance(){  
    double ponderacionAcum = 0;  
    for (Nota nota : notas) {  
        ponderacionAcum+=nota.getPonderacion();  
    }  
    return ponderacionAcum;  
}
```

En la clase estudiante escribimos los siguientes atributos, un array de asignaturas, un int de cantidad de estudiantes, un string de nombre de la asignatura.

```
private Asignatura[] asignaturas;  
private int cantidadAsig;  
private String nombre;
```

Escribimos nuestro método constructor validando de que el nombre de la asignatura no este vacío

```
public Estudiante(String nombre) {  
    if(nombre.trim().equals("") || nombre == null){  
        throw new IllegalArgumentException("Nombre vacio");  
    }  
    this.nombre = nombre;  
    asignaturas = new Asignatura[10];  
    cantidadAsig = 0;  
}
```

Generamos los métodos getNombre, getAsignatura

```
public Asignatura[] getAsignaturas() {  
    return asignaturas;  
}  
  
public String getNombre() {  
    return nombre;  
}
```

Creamos un método para crear asignatura

```
public void addAsignatura(Asignatura a){  
    asignaturas[cantidadAsig] = a;  
    cantidadAsig++;  
}
```

Agregamos métodos el método de calcular el promedio de la asignatura

```
public double promedio(){  
    double sumaPromedio = 0;  
    double sumaCredito = 0;  
    for (int i = 0; i < cantidadAsig; i++) {  
        sumaPromedio += asignaturas[i].promedio()*asignaturas[i].getCredito();  
        sumaCredito += asignaturas[i].getCredito();  
    }  
    return sumaPromedio/sumaCredito;  
}
```

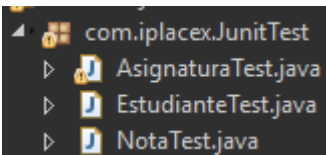
Creemos la clase nota con los siguientes atributos valor y ponderación

```
private double valor;  
private double ponderacion;
```

Creemos el método constructor de la clase con las excepciones

```
public Nota(double valor, double ponderacion) {  
    if(valor<=0){  
        throw new IllegalArgumentException("Valor mayor que 0");  
    }  
    if(ponderacion<=0){  
        throw new IllegalArgumentException("Ponderacion mayor que 0");  
    }  
    this.valor = valor;  
    this.ponderacion = ponderacion;  
}
```

Luego en el directorio src/test/java creamos la siguiente estructura que es la que se encargara de realizar las pruebas unitaria con junit con la siguiente estructura



```
com.iplacex.JUnitTest  
├── AsignaturaTest.java  
├── EstudianteTest.java  
└── NotaTest.java
```

En la clase AsignaturaTest importamos las librerías correspondientes

```
package com.iplacex.JunitTest;

import java.util.List;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

import entidad.Asignatura;
import entidad.Nota;
```

Dentro de la clase AsignaturaTest escribimos la siguiente estructura

```
public class AsignaturaTest {  
  
    public AsignaturaTest() {  
    }  
  
    @BeforeClass  
    public static void setUpClass() {  
    }  
  
    @AfterClass  
    public static void tearDownClass() {  
    }  
  
    @Before  
    public void setUp() {  
    }  
  
    @After  
    public void tearDown() {  
    }  
}
```

Luego escribimos el método con la anotación @Test

```
@Test  
public void testPromedio() {  
    System.out.println("Promedio");  
    try {  
        Asignatura instance = new Asignatura("Java", 2);  
        instance.getNotas().add(new Nota(4,0.2));  
        instance.getNotas().add(new Nota(6,0.2));  
        double resultado = 5;  
        assertEquals(instance.promedio(), resultado, 0);  
    } catch (IllegalArgumentException e) {  
        System.out.println(e.getMessage());  
        fail(e.getMessage());  
    }  
}
```

Escribimos el método del avance de la asignatura

```
@Test
public void testAvance() {
    System.out.println("Avance");
    try {
        Asignatura instance = new Asignatura("Java", 2);
        instance.getNotas().add(new Nota(4,0.2));
        instance.getNotas().add(new Nota(6,0.2));
        double resultado = 0.4;
        assertEquals(instance.avance(), resultado, 0);
    } catch (IllegalArgumentException e) {
        System.out.println(e.getMessage());
        fail(e.getMessage());
    }
}
```

Creamos la clase EstudianteTest

```
package com.iplacex.JunitTest;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

import entidad.Asignatura;
import entidad.Estudiante;
import entidad.Nota;

import static org.junit.Assert.*;
```

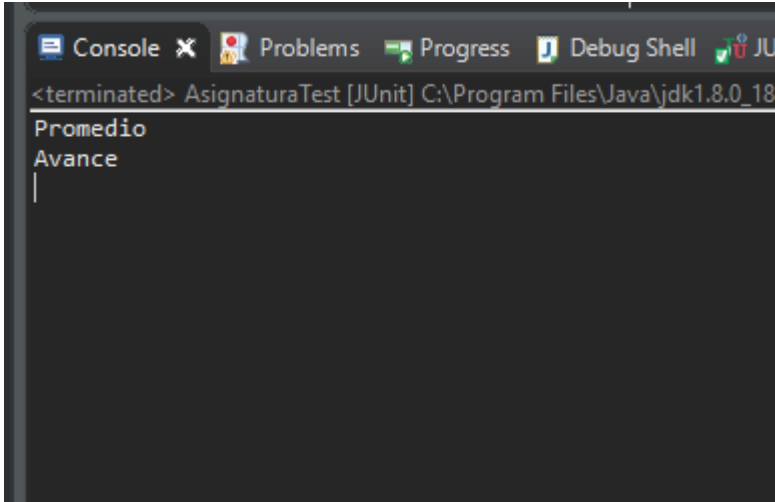
Seguimos la siguiente estructura


```
public EstudianteTest() {  
}  
  
@BeforeClass  
public static void setUpClass() {  
}  
  
@AfterClass  
public static void tearDownClass() {  
}  
  
@Before  
public void setUp() {  
}  
  
@After  
public void tearDown() {  
}
```

Escribimos el método que valida el promedio

```
@Test  
public void testPromedio() {  
    System.out.println("promedio");  
    try {  
        Estudiante instance = new Estudiante("Pepito");  
        Asignatura a1 = new Asignatura("Java", 4);  
        a1.getNotas().add(new Nota(4,0.2));  
        a1.getNotas().add(new Nota(6,0.2));  
        Asignatura a2 = new Asignatura("NET", 4);  
        a2.getNotas().add(new Nota(5,0.3));  
        a2.getNotas().add(new Nota(6,0.3));  
        instance.addAsignatura(a1);  
        instance.addAsignatura(a2);  
        double result = 5.25;  
        assertEquals(instance.promedio(), result, 0);  
    } catch (IllegalArgumentException e) {  
        System.out.println(e.getMessage());  
        fail(e.getMessage());  
    }  
}
```

Luego presionamos en el proyecto clic derecho y run as junit Test nos mostrara el siguiente resultado.



```
<terminated> AsignaturaTest [JUnit] C:\Program Files\Java\jdk1.8.0_181
Promedio
Avance
|
```

1.2 Crear un proyecto simple de Selenium para ejecutarlo en Jenkins como evidencia de configuración, realizando una prueba básica.

Creamos un nuevo proyecto llamado TestSeleniun

Luego una vez creara nuestro proyecto agregamos a nuestro archivo pom.xml la siguiente librerías

```
16
17 <dependencies>
18   <dependency>
19     <groupId>junit</groupId>
20     <artifactId>junit</artifactId>
21     <version>4.12</version>
22     <scope>test</scope>
23   </dependency>
24   <dependency>
25     <groupId>org.apache.maven.plugins</groupId>
26     <artifactId>maven-compiler-plugin</artifactId>
27     <version>3.8.1</version>
28   </dependency>
29   <dependency>
30     <groupId>org.seleniumhq.selenium</groupId>
31     <artifactId>selenium-java</artifactId>
32     <version>3.141.59</version>
33   </dependency>
34   <!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-surefire-plugin -->
35   <dependency>
36     <groupId>org.apache.maven.plugins</groupId>
37     <artifactId>maven-surefire-plugin</artifactId>
38     <version>3.0.0-M3</version>
39   </dependency>
40   <dependency>
41     <groupId>org.testng</groupId>
42     <artifactId>testng</artifactId>
43     <version>6.1.1</version>
44     <scope>compile</scope>
45   </dependency>
46 </dependencies>
```

Instalamos el plugin de testNG

Nota: Para la instalación de TestNG la versión compatible del entorno de desarrollo debe ser la versión 2018-9 para descargar deben al siguiente enlace y descargar la versión IDE for JAVA EE DEVELOPER

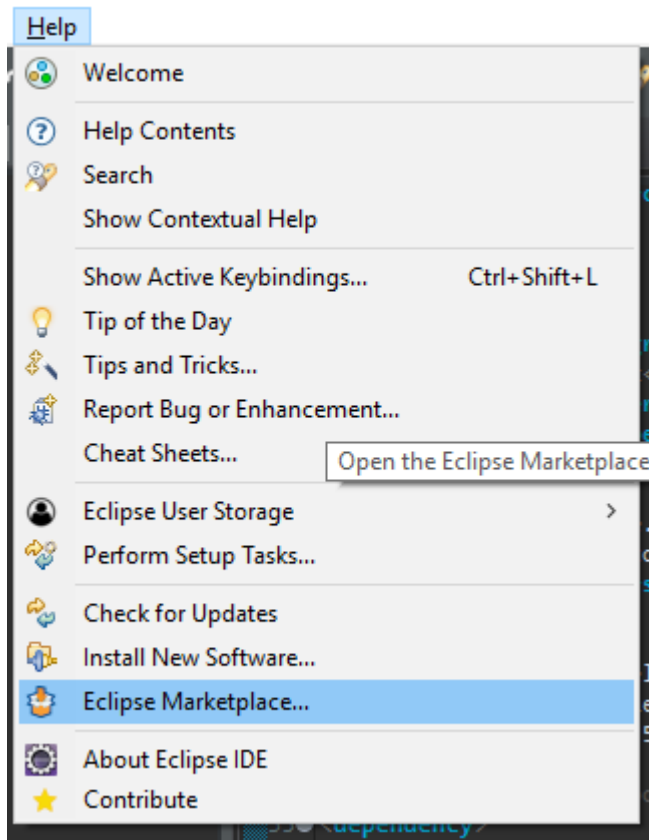
<https://www.eclipse.org/downloads/packages/release/2018-09/r>

Luego ir al siguiente enlace:

<https://marketplace.eclipse.org/content/testng-eclipse>

posteriormente instalar plugin en eclipse

abriendo la siguiente ventana



Después debemos arrastrar el botón de instalar dentro del Marketplace de eclipse

TestNG for Eclipse



☆ 561 💬 26



★ MPC DOWNLOADS
Top 10


Details

Screenshots

Metrics

Errors

External Install Button

This plug-in lets you run your TestNG tests from Eclipse. You can run suites, groups or individual methods. Errors are reported in a separate tab that lets you jump to failing tests efficiently. The plug-in also contains several templates to create tests easily.

Categories: [Testing](#)
Tags: [testng](#) [junit](#) [testing](#) [unit](#) [integration](#) [functional](#) [selenium](#)

Additional Details

Eclipse Versions:

Oxygen (4.7), Neon (4.6), Mars (4.5), Luna (4.4), Kepler (4.3), Juno (4.2, 3.8), Previous to Juno (<=4.1), Photon (4.8), 2018-09 (4.9)

Platform Support: Windows, Mac, Linux/GTK

Organization Name: [Cédric Beust](#)
Date Created: Thu, 2010-06-24 09:51

Development Status: Production/Stable

License: Apache 2.0

Date Updated: Mon, 2019-02-04 13:55

Submitted by: [Cedric Beust](#)

Thursday, July 11, 2019 - 02:09

Luego presionar en install

Eclipse Marketplace

Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.

Search Recent Popular Favorites Installed 2019 in Focus

Find: All Markets All Categories Go**Featured****TestNG for Eclipse**

This plug-in lets you run your TestNG tests from Eclipse. You can run suites, groups or individual methods. Errors are reported in a separate tab that lets you...
[more info](#)

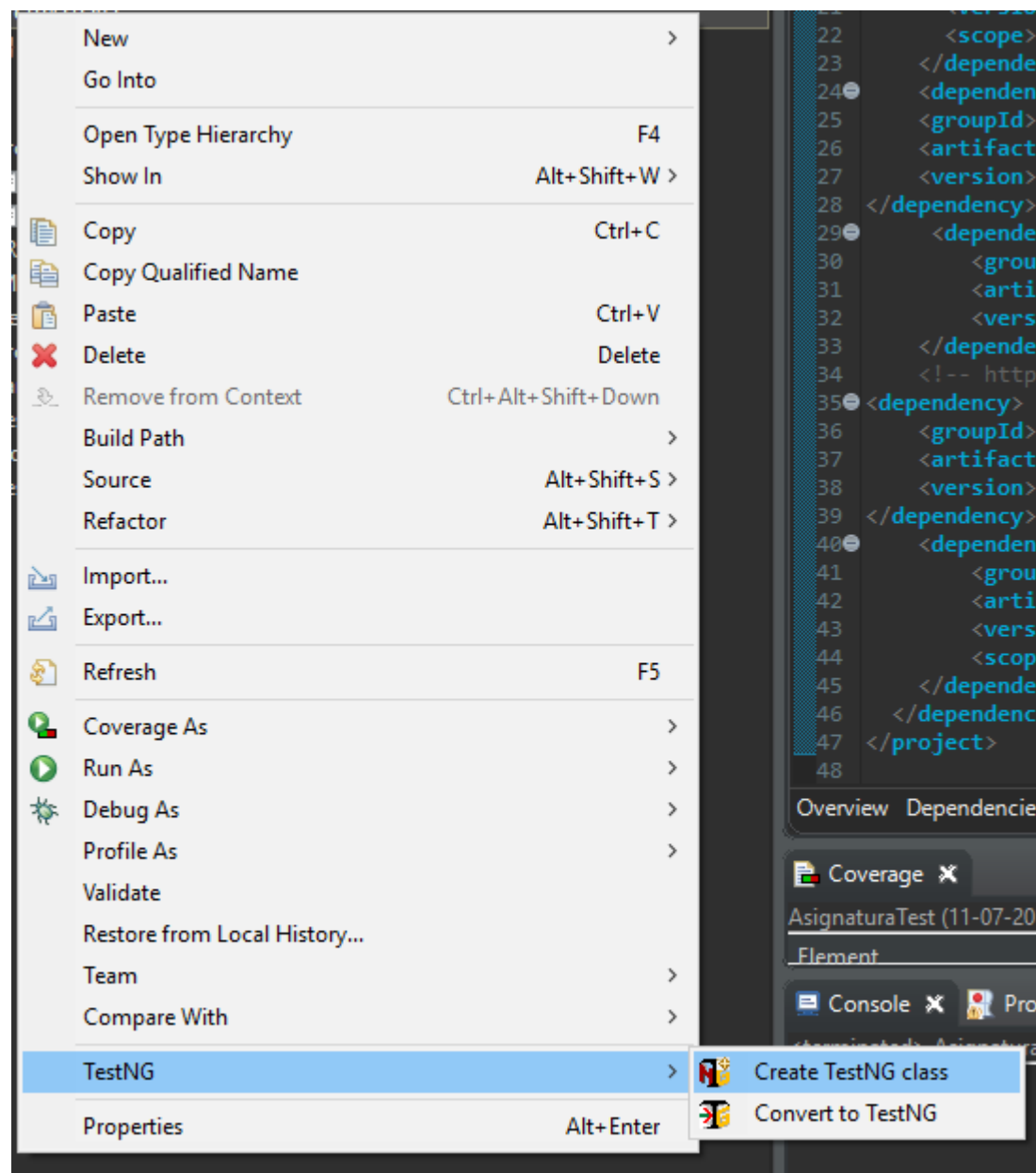
by [Cédric Beust](#) Apache 2.0[testng](#) [junit](#) [testing](#) [unit integration](#) [functional](#) [selenium](#)

★ 561

Installs: **885K** (13,544 last month)

Installed

Luego crearemos una clase en el directorio de tipo testNG en el directorio src/test/java



Importar la librerías que se utilizara incluyendo las de Selenium

```
1 package com.iplacex.SeleniumTest;
2
3
4
5 import static org.junit.Assert.assertEquals;
6
7 import java.util.concurrent.TimeUnit;
8
9 import org.openqa.selenium.By;
10 import org.openqa.selenium.WebDriver;
11 import org.openqa.selenium.WebElement;
12 import org.openqa.selenium.chrome.ChromeDriver;
13 import org.testng.annotations.AfterTest;
14 import org.testng.annotations.BeforeTest;
15 import org.testng.annotations.Test;
```

Posteriormente declaramos un atributo de tipo WebDriver llamado driver

```
1 private WebDriver driver;
2
3 @Test
```

Creamos el método f el cual realizara la prueba automatizada

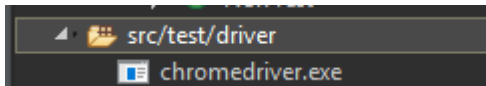
```
1 private WebDriver driver;
2
3 @Test
4 public void f() {
5     WebElement cuadroBusqueda = driver.findElement(By.name("q"));
6
7     cuadroBusqueda.sendKeys("pagina principal iplacex");
8     cuadroBusqueda.click();
9
10    cuadroBusqueda.submit();
11
12    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
13
14    assertEquals("pagina principal iplacex - Google search ", driver.getTitle());
15 }
16
```


Una vez creada esta estructura debemos agregar los drivers de Google Chrome la cual se encuentra en los recursos en el siguiente enlace:

<https://drive.google.com/drive/folders/190I-6IWCoD0VqF521aOvAWIxBIIKms5K>

Nota: esta versión del driver de Google Chorme funciona solamente con la versión del navegador 75.0.3770.90 de ser otra versión no funcionara

Agregar el driver de Google Chrome en el directorio a continuación



Luego crearemos el método beforeTes en el cual configuraremos las rutas de driver de Google Chrome

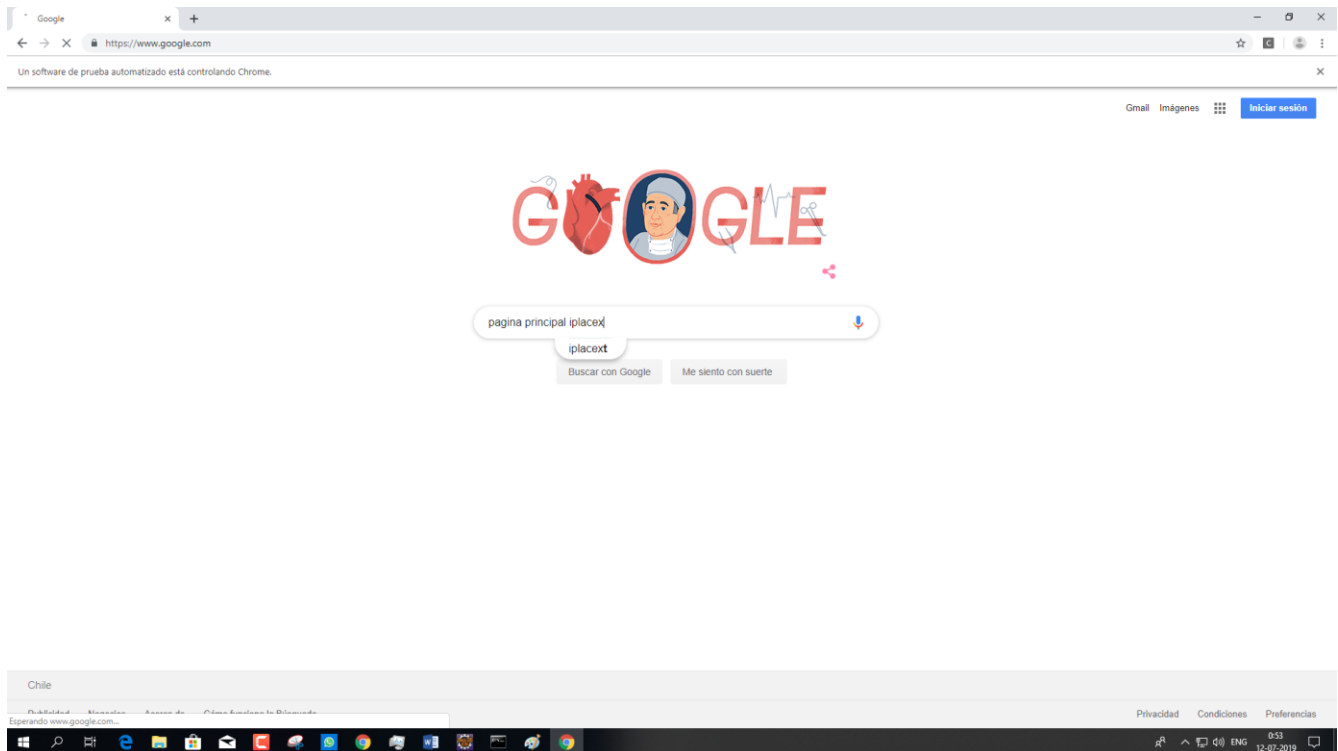
```
@BeforeTest
public void beforeTest() {

    System.setProperty("webdriver.chrome.driver", "C:\\Users\\Iplacex-PC\\Downloads\\ChromeDriver\\chromedriver.exe");
    driver = new ChromeDriver();
    driver.manage().window().maximize();
    driver.get("https://www.google.com/");
}
```

Luego crear el método de tipo void afterTest el cual cierrara el test.

```
@AfterTest
public void afterTest() {
    driver.quit();
}
```

Ejecutamos la prueba y posteriormente se abrirá el navegador Google Chrome realizando la búsqueda de forma automática



Un software de prueba automatizado está controlando Chrome.

Google

pagina principal iplacex

Cerca de 16.000 resultados (0.22 segundos)

Iplacex Aula Virtual
www.iplacex.cl/online/aula-virtual
Instituto Profesional Iplacex aula virtual. Portal del alumno en línea para de carreras online, profesionales y perfeccionamiento.
Portal de Alumnos en Línea · Pagos Online · Biblioteca · Validación de Certificados

¿Por qué estudiar a través de e-learning Iplacex?
www.iplacex.cl/res/
Posee una red de Centros de atención de alumnos en Valparaíso, Concepción y Punta Arenas y una red de oficinas que se presta soporte a las actividades de ...
Falta(n): principal

Carreras IPLACEX
www.iplacex.cl/res/carreras
Escoge la carrera que más se adecuó a ti! Tanto como carreras profesionales como también técnicas.
Matriculas abiertas!
Falta(n): principal
Carreras Profesionales · Carreras Técnicas · Técnico en Administración de ...

Alumnos Online - Iplacex
www.iplacex.cl/res/alumnos
Estudiar on line en Chile en 2017: admisión, matrículas, calendarios e instructivos para carreras técnicas y profesionales de Instituto Profesional Iplacex.

Sedes | Iplacex
www.iplacex.cl/res/sedes-ced-online
Su principal centro de operaciones se ubica en la ciudad de Santiago, donde se localizan: El centro de servicios de tutorías, La unidad de desarrollo curricular y ...

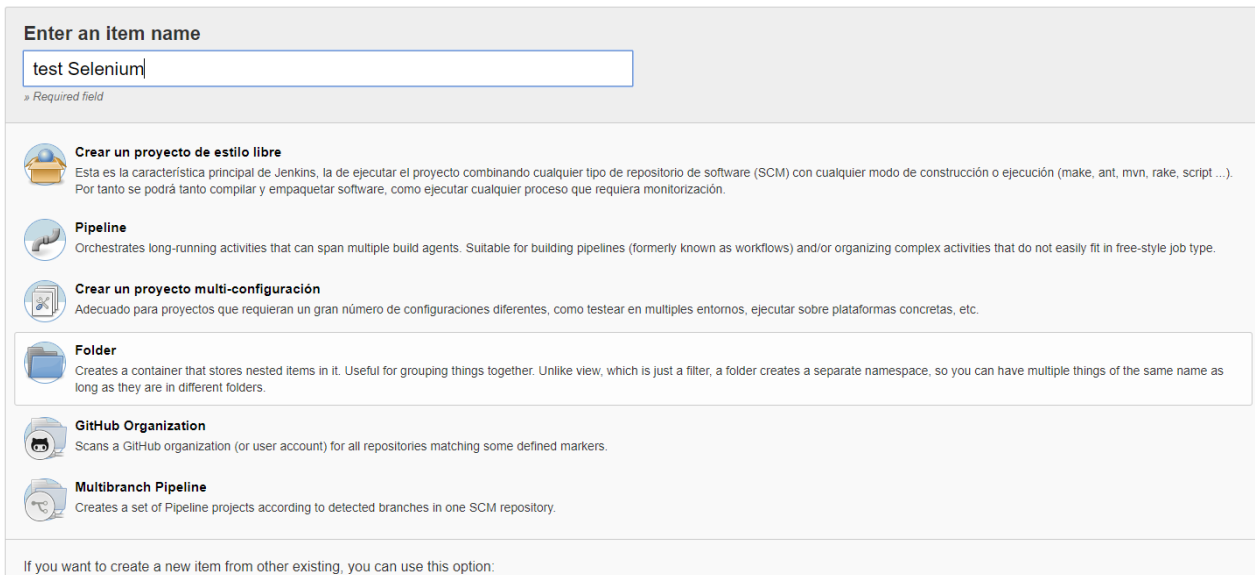
Plataforma virtual - Iplacex
https://ced.iplacex.cl/
Saltar a contenido principal. Plataforma virtual · Acceder · Biblioteca · Biblioteca VirtualSistema De Biblioteca-Servicios. Tutores. Modalidad ... Cancelar: Iplacex ...

Esperando www.google.com...

Windows taskbar: 0:54 12-07-2019

Programar una nueva tarea en Jenkins

Paso 1 ingresamos nuestro sitio de Jenkins, creamos una nueva tarea de proyecto estilo libre de nombre test selenium



Enter an item name

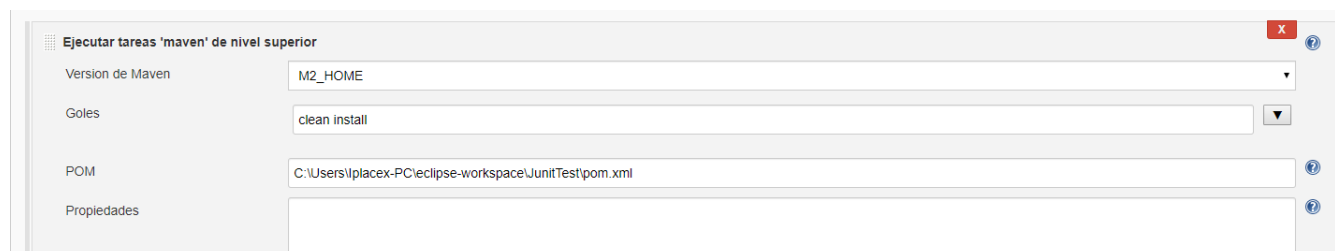
test Selenium

» Required field

- Crear un proyecto de estilo libre**
Esta es la característica principal de Jenkins, la de ejecutar el proyecto combinando cualquier tipo de repositorio de software (SCM) con cualquier modo de construcción o ejecución (make, ant, mvn, rake, script...). Por tanto se podrá tanto compilar y empaquetar software, como ejecutar cualquier proceso que requiera monitorización.
- Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Crear un proyecto multi-configuración**
Adecuado para proyectos que requieran un gran número de configuraciones diferentes, como testear en multiples entornos, ejecutar sobre plataformas concretas, etc.
- Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- GitHub Organization**
Scans a GitHub organization (or user account) for all repositories matching some defined markers.
- Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

If you want to create a new item from other existing, you can use this option:

En el Recuadro de ejecución ingresamos la ruta en donde se encuentra guardado nuestro archivo pom .xml



Ejecutar tareas 'maven' de nivel superior

Version de Maven: M2_HOME

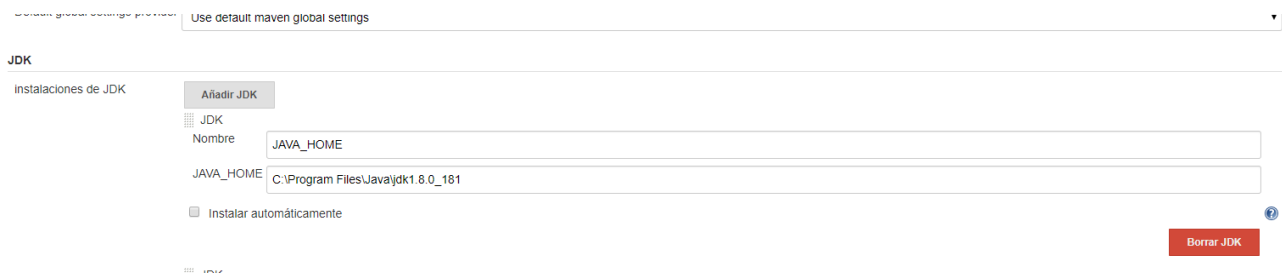
Goals: clean install

POM: C:\Users\lplacex-PC\workspace\JUnitTest\pom.xml

Propiedades:

Nota en caso de que al momento de realizar la prueba aparezca un error de jdk se debe agregar las rutas instaladas en el equipo de forma local

Como muestra en la siguiente imagen ir a administración de jenkins->Global tools configuración seleccionar la opción de jdk y desmarcar la opción de instalación automática.



Global tool configuration

Use default maven global settings

JDK

Instalaciones de JDK

Añadir JDK

JDK

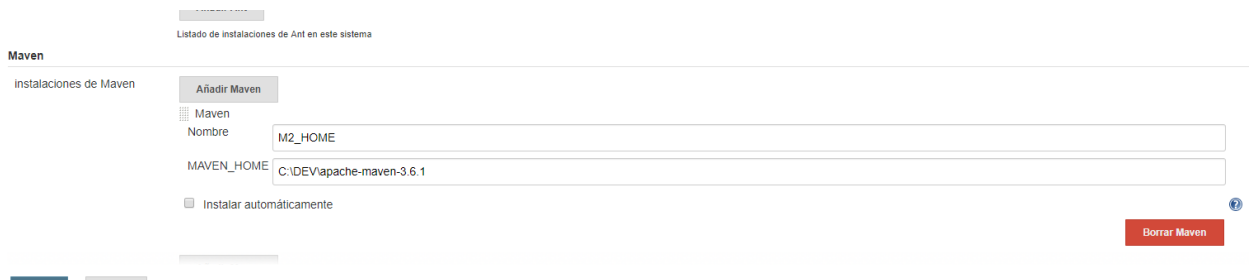
Nombre

JAVA_HOME

☐ Instalar automáticamente

Borrar JDK

Repetir el mismo paso con maven.



Global tool configuration

Listado de instalaciones de Ant en este sistema

Maven

Instalaciones de Maven

Añadir Maven

Maven

Nombre

MAVEN_HOME

☐ Instalar automáticamente

Borrar Maven

Como ultimo paso para programar la tarea nos debemos ir a las configuraciones de la tarea en el panel de configuración en el recuadro de disparadores de ejecución

Disparadores de ejecuciones

- ☐ Lanzar ejecuciones remotas (ejem: desde 'scripts') ?
- ☐ Construir tras otros proyectos ?
- ☐ Consultar repositorio (SCM) ?
- ☒ Ejecutar periódicamente ?

Programador

⚠ Spread load evenly by using 'H 01 * * *' rather than '30 01 * * *'
Would last have run at viernes 12 de julio de 2019 01:30:28 Hora de Chile; would next run at sábado 13 de julio de 2019 01:30:28 Hora de Chile.

☐ GitHub hook trigger for GITScm polling ?

Aparecerá el resultado de la prueba ya automatizada el resultado en la consola de la tarea

Salida de consola

```
Ejecutado por el programador
Running as SYSTEM
Ejecutando en el espacio de trabajo C:\Users\Iplacex-PC\.jenkins\workspace\clean install
[clean install] $ cmd.exe /C "C:\DEV\apache-maven-3.6.1\bin\mvn.cmd -f C:\Users\Iplacex-PC\eclipse-workspace\JUnitTest\pom.xml clean install && exit %%ERRORLEVEL%%"
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.iplacex:JUnitTest >-----
[INFO] Building JUnitTest 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ JUnitTest ---
[INFO] Deleting C:\Users\Iplacex-PC\eclipse-workspace\JUnitTest\target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ JUnitTest ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\Iplacex-PC\eclipse-workspace\JUnitTest\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ JUnitTest ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 3 source files to C:\Users\Iplacex-PC\eclipse-workspace\JUnitTest\target\classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ JUnitTest ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\Iplacex-PC\eclipse-workspace\JUnitTest\src\test\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ JUnitTest ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 3 source files to C:\Users\Iplacex-PC\eclipse-workspace\JUnitTest\target\test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ JUnitTest ---
[INFO] Surefire report directory: C:\Users\Iplacex-PC\eclipse-workspace\JUnitTest\target\surefire-reports

-----
T E S T S
-----
```

RESPUESTAS ESPERADAS

1. Respecto a Selenium herramienta para registrar acciones, permitiendo editarlas manualmente o crearlas desde cero. ¿Cuál es el potencial en el desarrollo de software basado en integración continua?

Se basan en el uso de diferentes API's en diferentes lenguajes, como: PHP, Ruby, JAVA, Javascript, etc. Entre su principales características podemos destacar: Facilidad de registro y ejecución de los test, Referencia a objetos DOM en base al ID, nombre o a través de Xpath, Auto-completado para todos los comandos, Las acciones pueden ser ejecutadas paso a paso, Herramientas de depuración y puntos de ruptura (breakpoints), Los test pueden ser almacenados en diferentes formatos.

2. ¿Cuales son las ventajas de Maven como sistema de automatización en pruebas funcionales?

Maven es una herramienta de automatización de compilación para proyectos de Java. Junto a Jenkins se pueden usar para desencadenar compilaciones continuas que incluyen, por ejemplo: la ejecución de pruebas JUnit cada vez que se compromete un nuevo código, el despliegue de estas construcciones para producción y la programación de estas tareas en momentos estratégicos del día, por ejemplo, medianoche. Estas herramientas y procesos conforman un enfoque DevOps para el desarrollo y la implementación de software y son populares en el desarrollo ágil.

3. ¿Cuales son los beneficios de la integración de Maven con Selenium en Jenkins?

La integración de Maven con Selenium proporciona los siguientes beneficios: Apache Maven brinda asistencia para administrar el ciclo de vida completo de un proyecto de prueba, Maven se utiliza para definir la estructura del proyecto, las dependencias, la compilación y la gestión de pruebas, Usando pom.xml (Maven) puede configurar las dependencias necesarias para

construir pruebas y ejecutar código, Maven descarga automáticamente los archivos necesarios desde el repositorio mientras construye el proyecto.