EBook Gratis

APRENDIZAJE testng

Free unaffiliated eBook created from
Stack Overflow contributors.

#testng

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: testng

It is an unofficial and free testng ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official testng.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con testng

Observaciones

Esta sección proporciona una descripción general de qué es testng y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema grande dentro de la prueba, y vincular a los temas relacionados. Dado que la Documentación para la prueba es nueva, es posible que deba crear versiones iniciales de los temas relacionados.

Versiones

Versión	Fecha
1.0	2017-06-07

Examples

Instalación o configuración

TestNG requiere JDK 7 o superior para usar.

De acuerdo con http://testng.org/doc/download.html para instalar testng, debe agregar la dependencia de testng a su archivo pom.xml o gradle build.gradle de maven

Maven

```
<repositories>
  <repository>
    <id>jcenter</id>
    <name>bintray</name>
    <url>http://jcenter.bintray.com</url>
  </repository>
</repositories>

<dependency>
  <groupId>org.testng</groupId>
  <artifactId>testng</artifactId>
  <version>6.9.12</version>
  <scope>test</scope>
</dependency>
```

Gradle:

```
repositories {
    jcenter()
```

```
}

dependencies {
    testCompile 'org.testng:testng:6.9.12'
}
```

Más opciones se pueden encontrar en la página oficial .

Programa rápido utilizando TestNG

```
package example;

import org.testng.annotations.*; // using TestNG annotations

public class Test {

 @BeforeClass
 public void setUp() {
   // code that will be invoked when this test is instantiated
 }

 @Test(groups = { "fast" })
 public void aFastTest() {
   System.out.println("Fast test");
 }

 @Test(groups = { "slow" })
 public void aSlowTest() {
    System.out.println("Slow test");
 }

}
```

El método `setUp()` se invocará después de que se haya construido la clase de prueba y antes de que se ejecute cualquier método de prueba. En este ejemplo, estaremos corriendo el grupo rápido, por lo `aFastTest()` será invocado mientras `aSlowTest()` se omitirán.

Ejemplo de prueba de Hello World

Escribir y ejecutar un programa simple de `TestNG` es principalmente un proceso de 3 pasos.

1. Código: escriba la lógica de negocios de su prueba y anótela con anotaciones de TestNG
2. Configurar: agregue información de su prueba en `testng.xml` o en `build.xml`
3. Ejecute TestNG : se puede invocar desde la línea de comandos, ANT, IDE como Eclipse, IDEA de IntelliJ)

Breve explicación del ejemplo (lo que necesita ser probado) :

Tenemos una clase `RandomNumberGenerator` que tiene un método `generateFourDigitPin` que genera un PIN de 4 dígitos y devuelve como `int` . Así que aquí queremos probar si ese número aleatorio es de 4 dígitos o no. A continuación se muestra el código:

Clase a probar :

```
package example.helloworld;

public class RandomNumberGenerator {

public int generateFourDigitPin(){
    return (int)(Math.random() * 10000);
}
}
```

La clase de prueba TestNG :

```
package example.helloworld;

import org.testng.Assert;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class TestRandomNumberGenerator {

    RandomNumberGenerator rng = null;

    @BeforeClass
    public void deSetup(){
        rng = new RandomNumberGenerator();
    }

    @Test
    public void testGenerateFourDigitPin(){
        int randomNumber = rng.generateFourDigitPin();
        Assert.assertEquals(4, String.valueOf(randomNumber).length());
    }

    @AfterClass
    public void doCleanup(){
        //cleanup stuff goes here
    }
}
```

Hay testng.xml :

```
<suite name="Hello World">
    <test name="Random Number Generator Test">
        <classes>
            <class name="example.helloworld.TestRandomNumberGenerator" />
        </classes>
    </test>
</suite>
```

Ejecutar TestNG suite con Gradle

Ejemplo de archivo `build.gradle` :

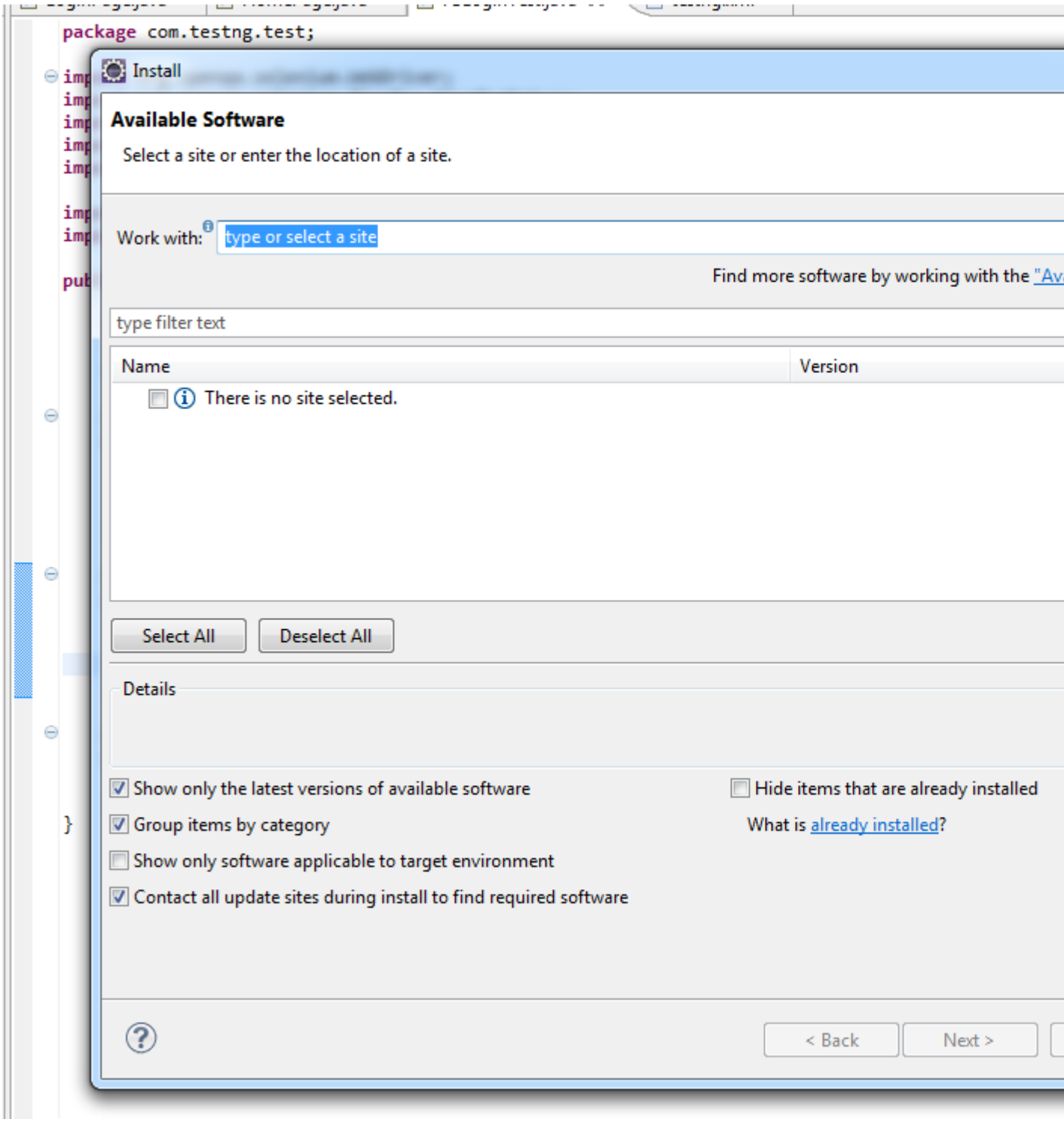
```
plugin: 'java'

repositories {
```

```
    mavenLocal()
    mavenCentral()
    jcenter()
}

dependencies {
    compile "org.testng:testng:6.9.12"
}

test {
    useTestNG() {
    suiteXmlBuilder().suite(name: 'Sample Suite') {
        test(name : 'Sample Test') {
            classes('') {
                'class'(name: 'your.sample.TestClass')
            }
        }
    }
}
```

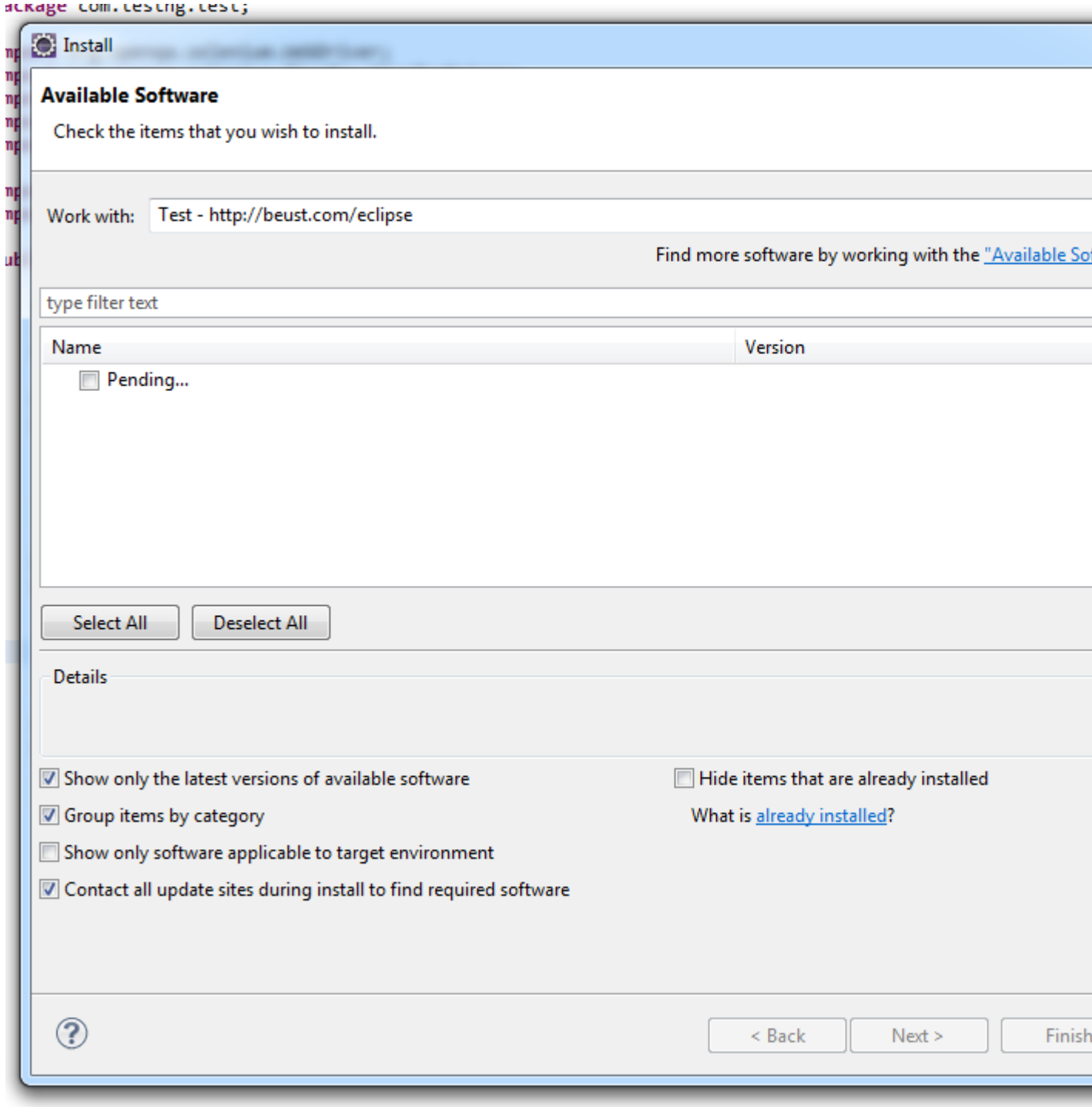
Cómo configurar TestNG en Eclipse & Run test usando xml

Cómo instalar TestNG en eclipse

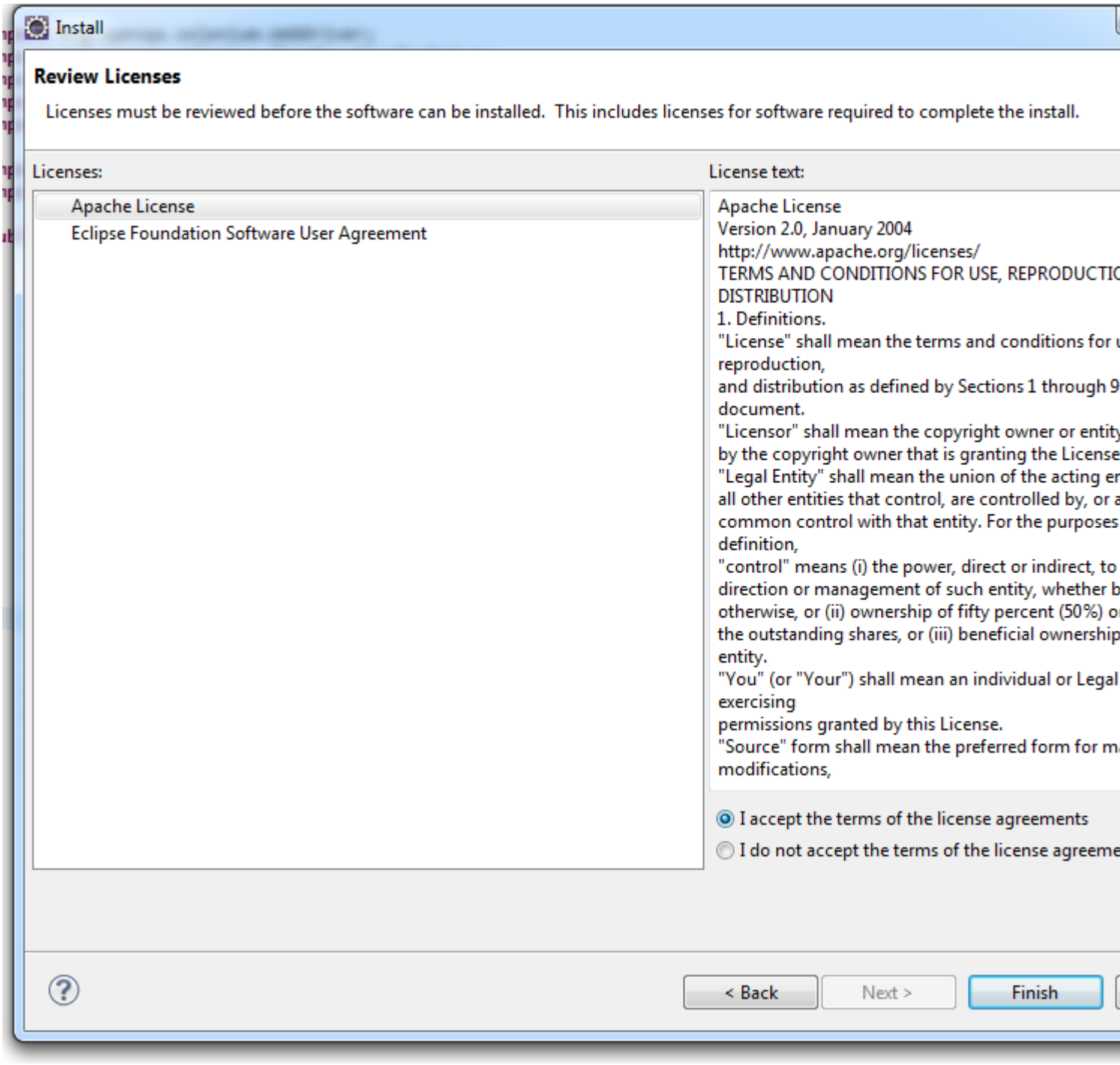
1. Eclipse abierto
2. Haga clic en Ayuda> Instalar nuevo software



3. Haga clic en Agregar
4. Proporcione nombre y URL - http://beust.com/eclipse



5. Seleccione TestNG
6. Haga clic en Siguiente



7. Haga clic en Finalizar
8. Tomará algún tiempo instalar TestNG

Una vez instalado, reinicie eclipse.

Permite crear un proyecto TestNG

1. Archivo> Nuevo> Proyecto Java> Proporcione un nombre y haga clic en Finalizar

2. Crear una clase como TestNGClass

3. Crear la siguiente clase

    1.LoginPage.class

2.HomePage.class

3.FBLoginTest.class

Aquí va el código:

Clase LoginPage

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

public class LoginPage {

    @FindBy(id = "email")
    private WebElement username;

    @FindBy(id = "pass")
    private WebElement password;

    @FindBy(xpath = ".//input[@data-testid='royal_login_button']")
    private WebElement login;

    WebDriver driver;

    public LoginPage(WebDriver driver){
        this.driver = driver;
         PageFactory.initElements(driver, this);
    }
    public void enterUserName(String name){
        username.clear();
        username.sendKeys(name);
    }

    public void enterPassword(String passwrd){
        password.clear();
        password.sendKeys(passwrd);
    }


    public HomePage clickLoginButton(){
        login.click();
        return new HomePage(driver);
    }
}
```

Clase de página de inicio .

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

public class HomePage {

    @FindBy(id = "userNavigationLabel")
    private WebElement userDropdown;
```

```
    WebDriver driver;

    public HomePage(WebDriver driver){
        this.driver = driver;
        PageFactory.initElements(driver, this);
    }

    public boolean isUserLoggedIn(){
        return userDropdown.isDisplayed();
    }

}
```

Clase FBLoginTest

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.Assert;
import org.testng.annotations.Test;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.AfterClass;

import com.testng.pages.HomePage;
import com.testng.pages.LoginPage;

public class FBLoginTest {

    WebDriver driver;
    LoginPage loginPage;
    HomePage homePage;

    @BeforeClass
    public void openFBPage(){
        driver = new FirefoxDriver();
        driver.get("https://www.facebook.com/");
        loginPage = new LoginPage(driver);
    }

    @Test
    public void loginToFB(){
        loginPage.enterUserName("");
        loginPage.enterPassword("");
        homePage = loginPage.clickLoginButton();
        Assert.assertTrue(homePage.isUserLoggedIn());
    }

    @AfterClass
    public void closeBrowser(){
        driver.quit();
    }

}
```

Aquí viene el testng xml: Haga clic derecho en Proyecto para crear un archivo xml y copie y pegue este contenido.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<suite name="Suite">
  <test name="Test">
    <classes>
      <class name="com.testng.FBLoginTest"/>
    </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->
```

Cómo agregar jarra de selenio independiente:
Descargue el último tarro independiente de selenio y agregue eso en la ruta de compilación del proyecto.

1. Haga clic con el botón derecho en Proyecto> Crear ruta> Configurar ruta de construcción> Seleccionar bibliotecas> Agregar archivos externos

¿Cómo ejecutar el testNG xml? Haga clic derecho en el xml> Ejecutar como> TestNGSuite

Feliz codificación :)

Lea Empezando con testng en línea: https://riptutorial.com/es/testng/topic/5393/empezando-con-testng

Capítulo 2: @ Test Annotation

Sintaxis

- @Prueba
- @Test (attribute1 = attributeValue, atrribute2 = attributeValue, etc.)

Parámetros

Parámetro	Detalles
siempre corre	Si se establece en verdadero, este método de prueba siempre se ejecutará incluso si depende de un método que falló.
proveedor de datos	El nombre del proveedor de datos para este método de prueba.
dataProviderClass	La clase donde buscar el proveedor de datos. Si no se especifica, el proveedor de datos se buscará en la clase del método de prueba actual o en una de sus clases básicas. Si se especifica este atributo, el método del proveedor de datos debe ser estático en la clase especificada.
Depende de grupos	La lista de grupos de los que depende este método.
Depende de métodos	La lista de métodos de los que depende este método.
descripción	La descripción de este método.
habilitado	Si los métodos en esta clase / método están habilitados.
expectedExceptions	La lista de excepciones que se espera que arroje un método de prueba. Si no se lanza una excepción o una diferente a una en esta lista, esta prueba se marcará como falla.
grupos	La lista de grupos a los que pertenece esta clase / método.
invocationCount	El número de veces que este método debe ser invocado.
invocationTimeOut	El número máximo de milisegundos que esta prueba debe tomar durante el tiempo acumulado de todas las cuentas de invocación. Este atributo se ignorará si no se especifica invocationCount.
prioridad	La prioridad para este método de prueba. Las prioridades más bajas se programarán primero.
exitoso porcentaje	El porcentaje de éxito esperado de este método.

Parámetro	Detalles
singlehilos	Si se establece en verdadero, se garantiza que todos los métodos en esta clase de prueba se ejecutarán en el mismo subproceso, incluso si las pruebas se están ejecutando actualmente con `parallel="methods"` . Este atributo solo se puede usar en el nivel de clase y se ignorará si se usa en el nivel de método. **Nota** : este atributo solía llamarse secuencial (ahora en desuso).
se acabó el tiempo	El número máximo de milisegundos que debe tomar esta prueba.
threadPoolSize	El tamaño de la agrupación de hilos para este método. El método se invocará desde varios subprocesos según lo especificado por invocationCount. **Nota** : este atributo se ignora si no se especifica invocationCount

Examples

Ejemplo rápido en @Test anotación

`@Test` anotación `@Test` se puede aplicar a cualquier **clase** o **método** . Esta anotación marca una clase o un método como parte de la prueba.

1. `@Test` en el nivel del método: marque el método anotado como método de prueba
2. `@Test` a nivel de clase
    - El efecto de una anotación `@Test` nivel de clase es hacer que todos los métodos públicos de la clase se conviertan en métodos de prueba, incluso si no están anotados.
    - `@Test` anotación `@Test` también se puede repetir en un método si desea agregar ciertos atributos.

Ejemplo de `@Test` a nivel de método :

```
import org.testng.annotations.Test;

public class TestClass1 {
    public void notTestMethod() {
    }

    @Test
    public void testMethod() {
    }
}
```

Ejemplo de `@Test` a nivel de clase :

```
import org.testng.annotations.Test;

@Test
public class TestClass2 {
```

```
    public void testMethod1() {
    }

    @Test
    public void testMethod2() {
    }
}
```

Lea @ Test Annotation en línea: https://riptutorial.com/es/testng/topic/6716/--test-annotation

Capítulo 3: Grupos de prueba

Sintaxis

- @Test (groups = {"group1", "group.regression"}, Depende de los grupos = {"group2", "group3"})

Examples

Configuración de Grupos de TestNG y ejemplo básico.

Los grupos se pueden configurar bajo el elemento `Suite` y / o `Test` de `testng.xml` . Todos los grupos que están marcados como incluidos en `tesng.xml` se considerarán para ejecución, se excluirá uno excluido. Si un método `@Test` tiene múltiples grupos y de esos grupos, si alguno de los grupos se excluye en `testng.xml` ese método `@Test` no se ejecutará.

A continuación se muestra la configuración típica de `testng.xml` en `Test` nivel de `Test` para grupos en ejecución:

```
<suite name="Suite World">
<test name="Test Name">
  <groups>
    <run>
      <include name="functest" />
      <exclude name="regtest" />
    </run>
  </groups>
  <classes>
    <class name="example.group.GroupTest"/>
  </classes>
</test>
</suite>
```

Y así se verá la clase de prueba:

```
package example.group;

import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class GroupTest {

    @BeforeClass
    public void deSetup(){
        //do configuration stuff here
    }

    @Test(groups = { "functest", "regtest" })
    public void testMethod1() {
    }
```

```
    @Test(groups = {"functest", "regtest"} )
    public void testMethod2() {
    }

    @Test(groups = { "functest" })
    public void testMethod3() {
    }

    @AfterClass
    public void cleanUp(){
        //do resource release and cleanup stuff here
    }
}
```

Al ejecutar esta clase `GroupTest TestNG` solo se `testMethod3()` .

Explicación:

- `<include name="functest" />` todos los métodos de prueba del grupo `functest` son elegibles para ejecutarse si no están excluidos por ningún otro grupo.
- `<exclude name="regtest" />` no se pueden ejecutar métodos de prueba del grupo `regtest` .
- `testMethod1()` y `testMethod2()` están en el grupo `regtest` , por lo que no se ejecutarán.
- `testMethod3()` está en el grupo `regtest` , por lo que se ejecutará.

TestNG MetaGroups - Grupos de grupos

TestNG permite definir grupos que pueden incluir otros grupos. Los MetaGrupos combinan lógicamente uno o más grupos y controlan la ejecución de los métodos `@Test` que pertenecen a esos grupos.

En el siguiente ejemplo hay varios métodos `@Test` que pertenecen a diferentes grupos. Pocos son específicos de una pila en particular y pocos son pruebas de regresión y aceptación. Aquí se pueden crear MetaGrupos. Vamos a elegir cualquiera de los dos **MetaGrupos** simples:

1. `allstack` : incluye los grupos `liux.jboss.oracle` y `aix.was.db2` y permite que todos los métodos de prueba que pertenecen a cualquiera de esos grupos se ejecuten juntos.
2. `systemtest` : incluye `allstack regression` grupos de `allstack` , `regression` y `acceptance` y permite que todos los métodos de prueba que pertenecen a cualquiera de esos grupos se ejecuten juntos.

configuración **testng.xml**

```
<suite name="Groups of Groups">
    <test name="MetaGroups Test">
        <groups>
            <!-- allstack group includes both liux.jboss.oracle and aix.was.db2 groups -->
            <define name="allstack">
                <include name="liux.jboss.oracle" />
                <include name="aix.was.db2" />
            </define>

            <!-- systemtest group includes all groups allstack, regression and acceptance -->
```

```
            <define name="systemtest">
                <include name="allstack" />
                <include name="regression" />
                <include name="acceptance" />
            </define>

            <run>
                <include name="systemtest" />
            </run>
        </groups>

        <classes>
            <class name="example.group.MetaGroupsTest" />
        </classes>
    </test>

</suite>
```

MetaGroupsTest clase

```
package example.group;

import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

public class MetaGroupsTest {

    @BeforeMethod
    public void beforeMethod(){
        //before method stuffs - setup
    }

    @Test(groups = { "liux.jboss.oracle", "acceptance" })
    public void testOnLinuxJbossOracleStack() {
        //your test logic goes here
    }

    @Test(groups = {"aix.was.db2", "regression"} )
    public void testOnAixWasDb2Stack() {
        //your test logic goes here
    }

    @Test(groups = "acceptance")
    public void testAcceptance() {
        //your test logic goes here
    }

    @Test(groups = "regression")
    public void testRegression(){
        //your test logic goes here
    }

    @AfterMethod
    public void afterMthod(){
        //after method stuffs - cleanup
    }
}
```

Lea Grupos de prueba en línea: https://riptutorial.com/es/testng/topic/5821/grupos-de-prueba

Capítulo 4: Pruebas parametrizadas

Examples

Proveedores de datos

Los proveedores de datos permiten crear múltiples entradas de prueba para ejecutarse dentro de una prueba. Consideremos una prueba que verifica que los números se duplican correctamente. Para crear un proveedor de datos, proporcione un método estático que devuelva el `Object[][]` o el `Iterator<Object[]>` (este último permite el cálculo `@DataProvider` de las entradas de prueba) anotado con la anotación `@DataProvider` , con el `name` propiedad como una cadena única que identifica proveedor.

```
import org.testng.annotations.DataProvider;

public class DoublingDataProvider {
    public final static String DOUBLING_DATA_PROVIDER = "doublingDataProvider";

    @DataProvider(name = DOUBLING_DATA_PROVIDER)
    public static Object[][] doubling() {
        return new Object[][]{
                new Object[]{1, 2},
                new Object[]{2, 4},
                new Object[]{3, 6}
        };
    }
}
```

En el caso anterior, cada `Object[]` representa un conjunto de datos para un solo caso de prueba: aquí se duplicará el número, seguido del valor esperado después de la duplicación.

Para utilizar el proveedor de datos, complete la propiedad `dataProvider` de la prueba con el nombre del proveedor. Si el método del proveedor se definió fuera de la clase de prueba o sus clases base, también debe especificar la propiedad `dataProviderClass` . El método de prueba debe tomar los parámetros correspondientes a los elementos de la descripción del caso de prueba; aquí hay dos pasos.

```
import org.testng.annotations.Test;

import static org.testng.Assert.assertEquals;

public class DoublingTest {

    @Test(dataProvider = DoublingDataProvider.DOUBLING_DATA_PROVIDER, dataProviderClass =
DoublingDataProvider.class)
    public void testDoubling(int number, int expectedResult) {
        assertEquals(number * 2, expectedResult);
    }
}
```

Lea Pruebas parametrizadas en línea: https://riptutorial.com/es/testng/topic/5684/pruebas-

parametrizadas

Capítulo 5: TestNG - Procedimiento de Ejecución

Examples

Procedimiento de ejecución de los métodos API de prueba de TestNG.

```
public class TestngAnnotation {
   // test case 1
   @Test
   public void testCase1() {
      System.out.println("in test case 1");
   }

   // test case 2
   @Test
   public void testCase2() {
      System.out.println("in test case 2");
   }

   @BeforeMethod
   public void beforeMethod() {
      System.out.println("in beforeMethod");
   }

   @AfterMethod
   public void afterMethod() {
      System.out.println("in afterMethod");
   }

   @BeforeClass
   public void beforeClass() {
      System.out.println("in beforeClass");
   }

   @AfterClass
   public void afterClass() {
      System.out.println("in afterClass");
   }

   @BeforeTest
   public void beforeTest() {
      System.out.println("in beforeTest");
   }

   @AfterTest
   public void afterTest() {
      System.out.println("in afterTest");
   }

   @BeforeSuite
   public void beforeSuite() {
      System.out.println("in beforeSuite");
   }
```

```
   @AfterSuite
   public void afterSuite() {
      System.out.println("in afterSuite");
   }

}
```

vamos a crear el archivo testng.xml en C:> WORKSPACE para ejecutar anotaciones.

```
<suite name="Suite1">
  <test name="test1">
    <classes>
       <class name="TestngAnnotation"/>
    </classes>
  </test>
</suite>
```

C: \ WORKSPACE> javac TestngAnnotation.java

Ahora ejecute el testng.xml, que ejecutará el caso de prueba definido en la clase de caso de prueba proporcionada.

```
in beforeSuite
in beforeTest
in beforeClass
in beforeMethod
in test case 1
in afterMethod
in beforeMethod
in test case 2
in afterMethod
in afterClass
in afterTest
in afterSuite

===============================================
Suite
Total tests run: 2, Failures: 0, Skips: 0
===============================================
```

El procedimiento de ejecución es el siguiente:

1. En primer lugar, el método **beforeSuite ()** se ejecuta solo una vez.
2. Por último, el método **afterSuite ()** se ejecuta solo una vez.
3. Incluso los métodos **beforeTest ()** , **beforeClass ()** , **afterClass ()** y **afterTest ()** se ejecutan solo una vez.
4. **El** método **beforeMethod ()** se ejecuta para cada caso de prueba pero antes de ejecutar el caso de prueba.
5. **El** método **afterMethod ()** se ejecuta para cada caso de prueba pero después de ejecutar el caso de prueba.
6. Entre **beforeMethod ()** y **afterMethod ()** , se ejecuta cada caso de prueba.

Lea TestNG - Procedimiento de Ejecución en línea:
https://riptutorial.com/es/testng/topic/7889/testng---procedimiento-de-ejecucion

Creditos

S. No	Capítulos	Contributors
1	Empezando con testng	Atul Dwivedi, Community, Idos, mackowski, RocketRaccoon, Sudha Velan
2	@ Test Annotation	Atul Dwivedi, Benoit
3	Grupos de prueba	Atul Dwivedi
4	Pruebas parametrizadas	Benoit, mszymborski
5	TestNG - Procedimiento de Ejecución	Shrikant