

SERENITY BDD con CUCUMBER

4

Implementando pasos y objetos II

(Tiempo ejecución 4 horas)

Recordemos....

Un modelo BDD está compuesto por la siguiente estructura de conexión

Feature: cada línea de un scenario mapea con un método en la clase .definition

Definition: cada método mapea con unos métodos en la clase .steps

Steps: para la construcción de los pasos es necesaria la definición de objetos en .pageobjects

Pageobjects: contiene las clases con la definición de los objetos

Este es nuestro requisito:

Historia de Usuario: Verificar el diligenciamiento de la pantalla “Popup Validation”.

Criterios de Aceptación:

- Verificar diligenciamiento exitoso.
- Verificar mensaje de validación para cada campo

url de prueba: <https://colorlib.com/polygon/metis/login.html>

Pasos para la ejecución de la prueba.

1. Autenticacion en colorlib

- Abrir navegador con la url de prueba
- Ingresar usuario demo
- Ingresar password demo
- Click en botón Sign in
- Verificar la Autenticación (label en home)

2. Ingresar a Funcionalidad Popup Validation

- Clic en Menu “Forms”
- Clic en submenú “Form Validation”
- Verificación : se presenta pantalla de la funcionalidad con título Popup Validation

3. Diligenciar Formulario Popup Validation

- Diligenciar todos los campos del formulario
- Clic en botón Validate

4. Verificar Respuesta Exitosa/Fallida

Objetivo:

Durante esta sesión vamos a enfocarnos en implementar el código requerido para las acciones 2, 3 y 4.



Acción : Ingresar a Funcionalidad Popup Validation

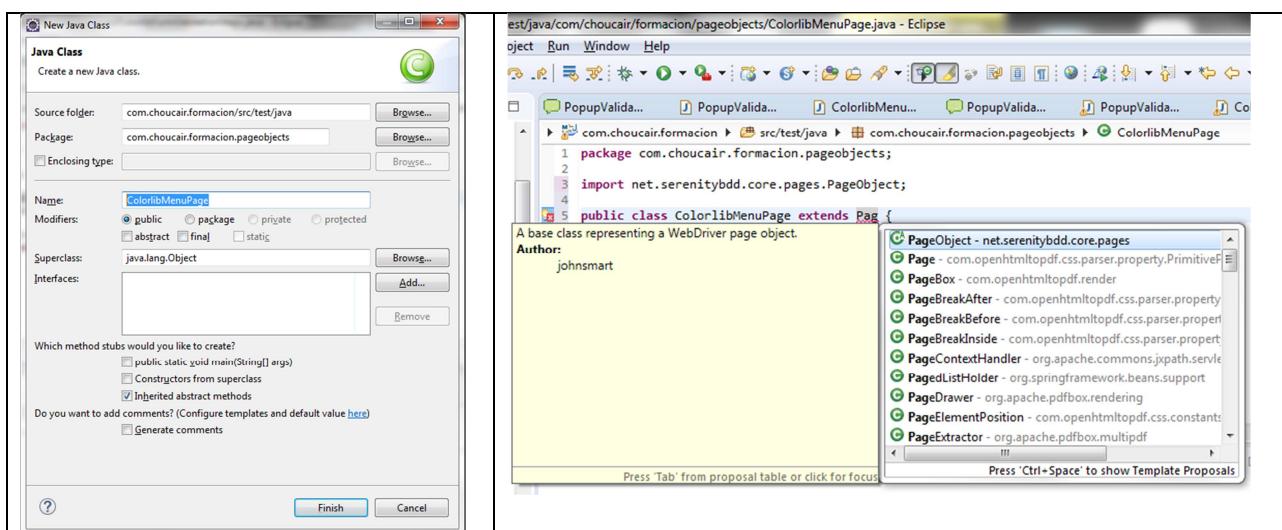
- Clic en Menu "Forms"
- Clic en submenú "Form Validation"
- Verificación : se presenta pantalla de la funcionalidad con título Popup Validation

Este ejercicio lo vamos a desarrollar cambiando el orden de construcción:

Paso 1, Crear los objetos, vamos a crear una clase en el paquete “*com.choucair.formacion.pageobjects*”, en la cual definiremos los objetos requeridos para interactuar con el menú y submenús.

Paso 1.1, crear la clase “*ColorlibMenuPage*” en el paquete “*com.choucair.formacion.pageobjects*”.

Paso 1.2, agregar el “**extends**” (Java permite el empleo de la herencia, característica muy potente que permite definir una clase tomando como base a otra clase ya existente.), si presionas Ctrl+<Space> después de escribir extends, el sistema te propondrá la opción de PageObject.



Paso 1.3, Definir los objetos, al igual que en la guía anterior debemos inspeccionar uno a uno los objetos requeridos para interactuar con el menú y definirlos en la clase.

Nota: pueden agregar la extensión “Ospy” a chrome la cual les ayudará a devolver las propiedades de los objetos

Los objetos a agregar serían:

Menú Forms

Submenú Form Validation

Y necesitamos un label que nos ayude a determinar si se cargó la pantalla requerida, para eso mapearemos el label Popup Validation (ver imagen)

Adicional vamos a mapear el submenú Form General



Paso 1.4, para efectos de reuso, vamos a crear métodos que realicen la acción de interactuar con las diferentes opciones de menú.

Paso 1.5, actualizar y agregar los import requeridos.

Al final tendríamos la siguiente clase:

```

1 package com.choucair.formacion.pageobjects;
2
3 import static org.hamcrest.MatcherAssert.assertThat;
4 import static org.hamcrest.Matchers.containsString;
5
6 import org.openqa.selenium.WebElement;
7
8 import net.serenitybdd.core.annotations.findby.FindBy;
9 import net.serenitybdd.core.pages.PageObject;
10
11 public class ColorlibMenuPage extends PageObject{
12     //Menu Forms
13     @FindBy(xpath="//*[@id='menu']/li[6]/a")
14     public WebElement menuForms;
15
16     //submenu Form General
17     @FindBy(xpath="//*[@id='menu']/li[6]/ul/li[1]/a")
18     public WebElement menuFormGenerals;
19
20     //submenu Form Validation
21     @FindBy(xpath="//*[@id='menu']/li[6]/ul/li[2]/a")
22     public WebElement menuFormValidation;
23
24     //Form Validation - label informativo
25     @FindBy(xpath="//*[@id='content']/div/div/div[1]/div/div/header/h5")
26     public WebElement lblFormValidation;
27
28     public void menuFormValidation() {
29         menuForms.click();
30         menuFormValidation.click();
31         String strMensaje = lblFormValidation.getText();
32         assertThat(strMensaje, containsString("Popup Validation"));
33     }
34 }
```

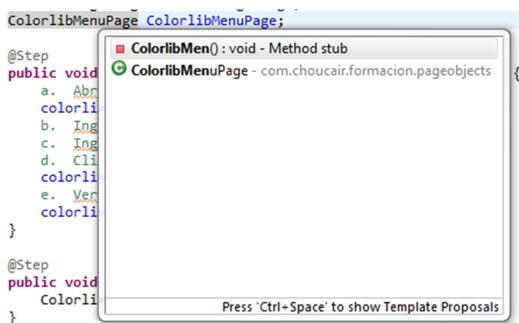
3.4

3.5



Paso 2, Agregar los **steps**, como en el paquete “com.choucair.formacion.steps” ya contamos con la clase “PopupValidationSteps”, vamos a agregar el método encargado de ejecutar el ingreso al menú Validation definido previamente en el pageobjects.

Paso 2.1, lo primero que debemos hacer es instanciar la clase “ColorlibMenuPage”, para esto ubíquese al inicio de la clase y escriba el nombre de la clase a instanciar, también puede apoyarse con **Ctrl + <space>** para obtener el nombre de una lista.



The screenshot shows a code editor with Java code. The cursor is at the end of a class definition. A code completion dropdown is open, showing suggestions for methods of the `ColorlibMenuPage` class. The suggestions include `ColorlibMenu()` (void - Method stub) and `ColorlibMenuPage` (com.choucair.formacion.pageobjects). The dropdown has a note at the bottom: `Press 'Ctrl+Space' to show Template Proposals`.

```

ColorlibMenuPage ColorlibMenuPage;
{
    @Step
    public void Abrir() {
        a. Abrir
        b. Ing
        c. Ing
        d. Cli
        colorlib
        e. Ver
        colorlib
    }

    @Step
    public void Colorlib() {
        Colorlib
    }
}

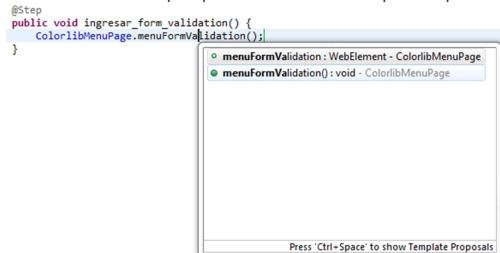
```

Paso 2.2, agregue un nuevo método con el que vamos a invocar el método creado previamente en la clase “ColorlibMenuPage”, recuerde utilizar la anotación `@Step` antes del método.

Tiene dos opciones para implementar el método steps

<pre> @Step public void ingresar_form_validation() { colorlibMenuPage.menuFormValidation(); } </pre>	Invocando el método menuFormValidation
<pre> @Step public void ingresar_form_validation() { colorlibMenuPage.menuFormValidation(); colorlibMenuPage.menuForms.click(); colorlibMenuPage.menuFormValidation.click(); String strMensaje = lblFormValidation.getText(); assertThat(strMensaje, containsString("Popup Validation")); } </pre>	Utilizando los objetos creados en el pageobject

Nota: recuerde que puede usar **Ctrl+<space>** para ver todos los métodos y variables disponibles de la clase extendida.



The screenshot shows a code editor with Java code. The cursor is at the end of a method call. A code completion dropdown is open, showing suggestions for methods of the `ColorlibMenuPage` class. The suggestions include `menuFormValidation` (WebElement - ColorlibMenuPage) and `menuFormValidation()` (void - ColorlibMenuPage). The dropdown has a note at the bottom: `Press 'Ctrl+Space' to show Template Proposals`.

```

@Step
public void ingresar_form_validation() {
    ColorlibMenuPage.menuFormValidation();
}

```



El resultado final sería así:

```

1 package com.choucair.formacion.steps;
2
3 import static org.hamcrest.MatcherAssert.assertThat;
4 import static org.hamcrest.Matchers.containsString;
5
6 import com.choucair.formacion.pageobjects.ColorlibLoginPage;
7 import com.choucair.formacion.pageobjects.ColorlibMenuPage;
8
9 import net.thucydides.core.annotations.Step;
10
11 public class PopupValidationSteps {
12
13     ColorlibLoginPage colorlibLoginPage;
14     ColorlibMenuPage colorlibMenuPage;
15
16     @Step
17     public void login_colorlib(String strUsuario, String strPass) {
18         // a. Abrir navegador con la url de prueba
19         colorlibLoginPage.open();
20         /* b. Ingresar usuario demo
21         c. Ingresar password demo
22         d. Click en botón Sign in */
23         colorlibLoginPage.IngresarDatos(strUsuario, strPass);
24         // e. Verificar la Autenticación (label en home)
25         colorlibLoginPage.VerificaHome();
26     }
27
28     @Step
29     public void ingresar_form_validation() {
30         colorlibMenuPage.menuFormValidation(); |
31     }
32 }
```

Paso 3, Mapear el método **definition** con los **steps**.

Ya tenemos la clase pero nuestro método en la definition no está conectado con los pasos.

<pre> @CasoExito Scenario: Diligenciamiento exitoso del formulario Popup Validation, no se presenta ningún mensaje de validación. Given Autentico en colorlib con usuario "demo" y clave "demo" And Ingreso a la funcionalidad Forms Validation When Diligencio Formulario Popup Validation Then Verifico ingreso exitoso </pre>	<pre> @Given("^Ingreso a la funcionalidad Forms Validation\$") public void ingreso_a_la_funcionalidad_Forms_Validation() { } </pre>
---	---

Paso 3.1, Abrimos la clase “PopupValidationDefinition”

Paso 3.2, editar el método agregando el llamado al método ingresar_form_validation() del paquete steps.

```

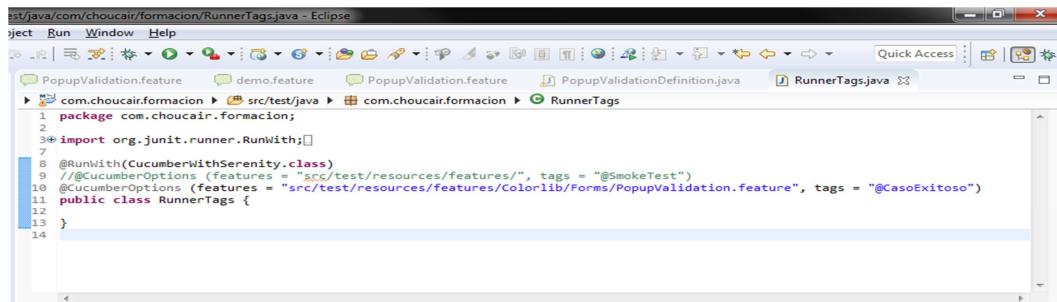
@Given("^Ingreso a la funcionalidad Forms Validation$")
public void ingreso_a_la_funcionalidad_Forms_Validation() {
    popupValidationSteps.ingresar_form_validation();
}

```



Paso 4, Ejecutar, abrimos la clase "RunnerTags" clic derecho sobre el runner > Run As > JUnit Test

Nota, ahora nuestro script además de autenticarse, deberá ingresar a la funcionalidad requerida y verifica que si nos encontremos en la pantalla respectiva.



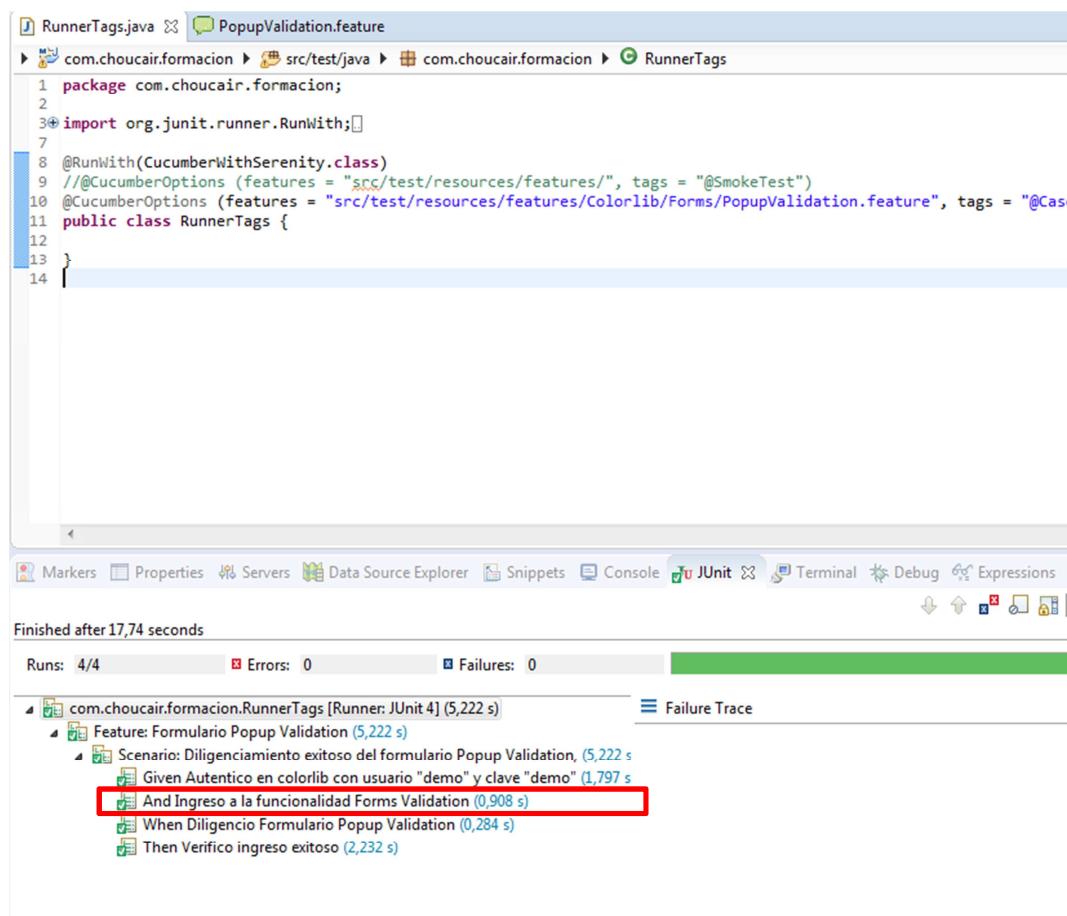
```

java/com/choucair/formacion/RunnerTags.java - Eclipse
Project Run Window Help
File Edit View Search Tools Window Help
PopupValidation.feature demo.feature PopupValidation.feature PopupValidationDefinition.java RunnerTags.java
com.choucair.formacion src/test/java com.choucair.formacion RunnerTags
1 package com.choucair.formacion;
2
3 import org.junit.runner.RunWith;
4
5 @RunWith(CucumberWithSerenity.class)
6 //@@CucumberOptions (features = "src/test/resources/features/", tags = "@SmokeTest")
7 @CucumberOptions (features = "src/test/resources/features/Colorlib/Forms/PopupValidation.feature", tags = "@CasoExitoso")
8 public class RunnerTags {
9
10
11 }
12
13
14

```

El sistema deberá abrir el navegador Chrome e ingresar a la aplicación colorlib y por último cerrar el navegador.

Dar clic sobre la pestaña Junit, en esta se visualiza el resultado de la ejecución para cada paso:



RunnerTags.java PopupValidation.feature

com.choucair.formacion src/test/java com.choucair.formacion RunnerTags

```

1 package com.choucair.formacion;
2
3 import org.junit.runner.RunWith;
4
5 @RunWith(CucumberWithSerenity.class)
6 //@@CucumberOptions (features = "src/test/resources/features/", tags = "@SmokeTest")
7 @CucumberOptions (features = "src/test/resources/features/Colorlib/Forms/PopupValidation.feature", tags = "@CasoExitoso")
8 public class RunnerTags {
9
10
11 }
12
13
14

```

Markers Properties Servers Data Source Explorer Snippets Console JUnit Terminal Debug Expressions

Finished after 17,74 seconds

Runs: 4/4 Errors: 0 Failures: 0

com.choucair.formacion.RunnerTags [Runner: JUnit 4] (5,222 s)

Feature: Formulario Popup Validation (5,222 s)

Scenario: Diligenciamiento exitoso del formulario Popup Validation, (5,222 s)

Given Autentico en colorlib con usuario "demo" y clave "demo" (1,797 s)

And Ingreso a la funcionalidad Forms Validation (0,908 s) And Ingreso a la funcionalidad Forms Validation (0,908 s)

When Diligencio Formulario Popup Validation (0,284 s)

Then Verifico ingreso exitoso (2,232 s)



Paso 8, generar evidencia

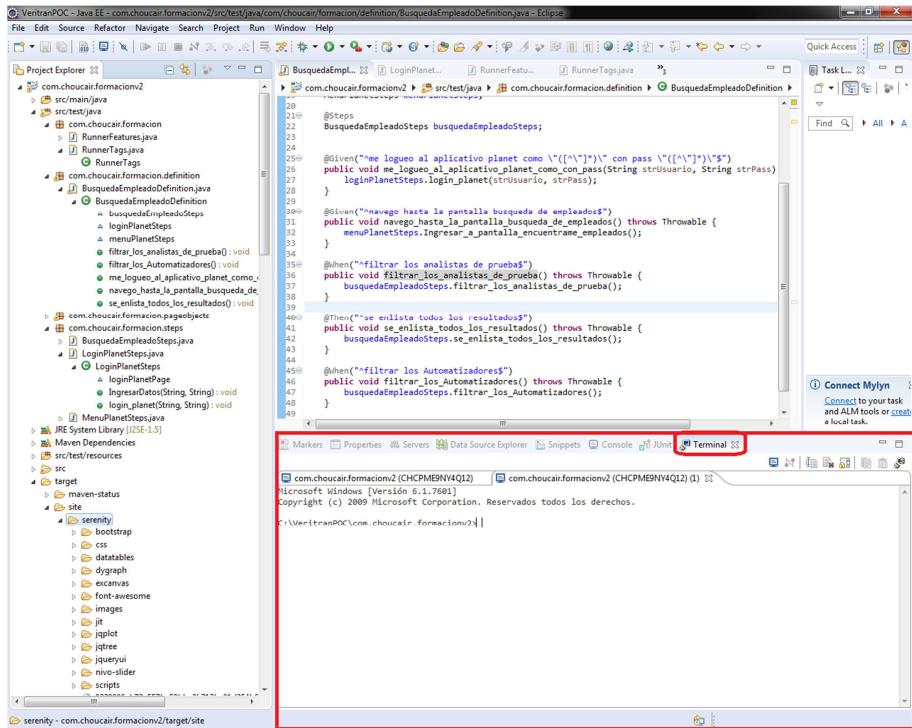
Generar Reporte de Ejecución

Click derecho sobre el proyecto > Show In > Terminal

Se activará la consola Terminal, así:

En esta se debe escribir el siguiente comando:

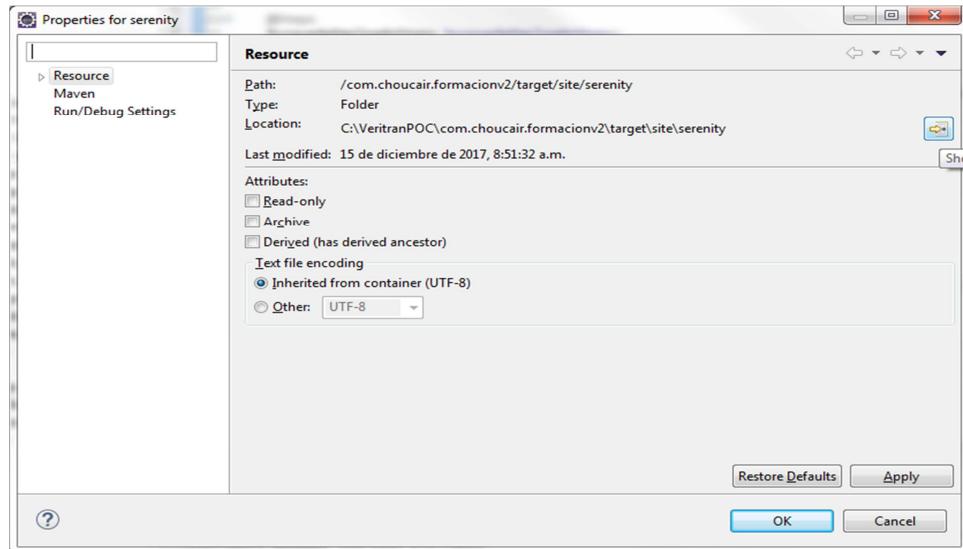
`mvn serenity:aggregate`



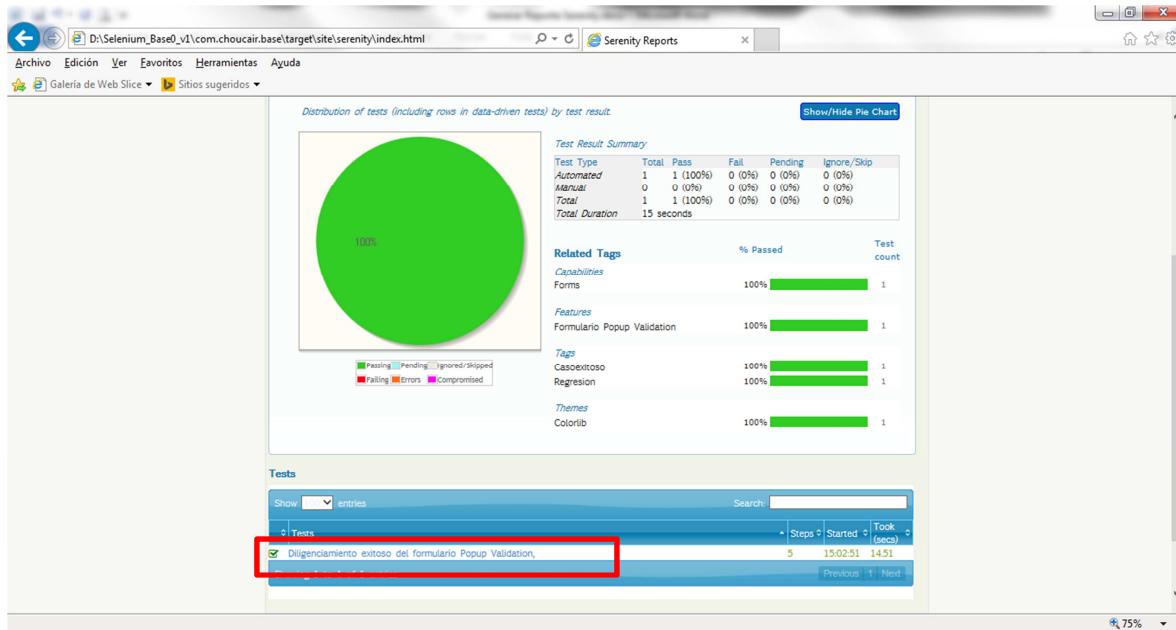
Una vez generado el reporte damos clic derecho al folder:

/target/site/serenity > Properties > Dar clic al botón "Show in system explorer"

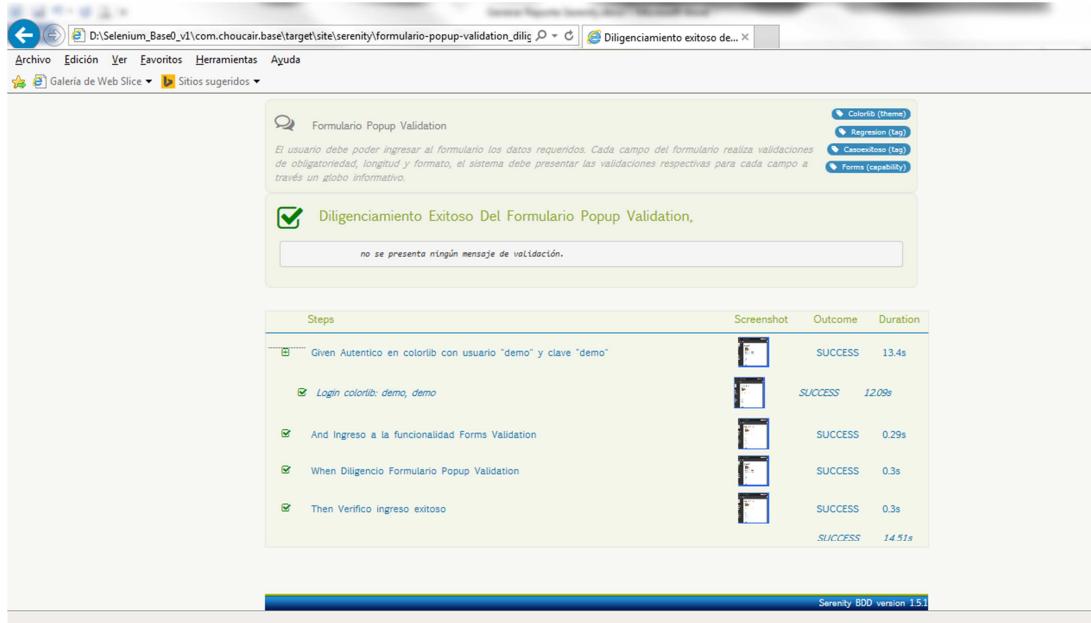




Ubicar en la carpeta el archive index.html y dar doble click.



Dar clic al test, se desplegarán los pasos ejecutados.



Ahora vamos a repetir el proceso para el diligenciamiento del formulario Popup Validation

Acción **Diligenciar Formulario Popup Validation**

- Diligenciar todos los campos del formulario
- Clic en botón Validate

En este caso vamos a requerir la data con la que vamos a diligenciar el formulario o DataDriven.

Existen varias formas de utilizar **datadriven** en **cucumber**, una de ellas es enviando los datos en la misma línea gherkin, como lo hicimos en la primera acción, donde los valores se envían entre comillas.

Given Autentico en colorlib con usuario "demo" y clave "demo"

Técnica de **Scenario Outline**, que consiste en una tabla adjunta en la historia y que puede ser utilizada por todos los escenarios y líneas gherkin existentes.

Ejemplo

```
Scenario Outline: Data driving new user sign-up
  And she provides the first name as <firstName>
  And she provides the last name as <lastName>
  And she provides the email as <email>
  And she provides the password as <password>
  And she provides the confirm password again as <password>
  And she signs-up
  Then she should be logged in to the application
Examples:
| firstName | lastName | email           | password |
| Sukesh   | Kumar    | validemail@aq.com | password |
```

Características:

- A la palabra “Scenario” debemos agregar “Outline”.
- Al final de todos los escenarios agregas un ítem “Examples:” y debajo una tabla separada por pipes(|) con una primer fila con los nombres de los campos y debajo de cada una de ellas los valores de prueba.
- Al interior de cada línea gherkin se puede hacer referencia a los valores con el nombre del campo **<firstName>**

Otra forma de usar los datos es con la técnica “**DataTable**”, la cual consiste en definir una tabla de datos para un paso o línea **gherkin** específica, la diferencia con el “**Outline**” es que estos datos sólo estarán disponibles para el paso en el cual definidos mientras que en el outline los datos pueden ser usados por todos los pasos del escenario.

Ejemplo de DataTable:

```
Scenario: Sign-up a new user with datatable example
  Given the user is on landing page
  When she chooses to sign up
  And she provides the her details as follows:
    | firstName | lastName | email           | password |
    | Sukesh   | Kumar    | validemail@aq.com | password |
  And she signs-up
  Then she should be logged in to the application
```



Para efectos de este taller vamos a usar la técnica **DataTable**:

Paso 1, identificar los datos requeridos por la pantalla a diligenciar y para cada uno de ellos agregar una columna.

Paso 1.1, Crear la tabla de datos, apóyese en un editor de texto para crear la tabla

Nota: los datos a crear deben ser válidos, puesto que estamos automatizando la ruta feliz.

Como el campo "Multiple Select" permite la selección múltiple de valores, vamos a crear dos columnas en el datatable para en determinado caso, poder enviar más de un valor.

Paso 1.2, una vez tengamos la tabla, la copiamos en la historia de usuario debajo del paso "Diligencio Formulario Popup Validation".

When Diligencio Formulario Popup Validation													
Required	Select	MultipleS1	MultipleS2	Url	Email	Password1	Password2	MinSize	MaxSize	Number	IP	Date	DateEarlier
Valor1	Golf	Tennis	Golf	http://www.valor1.com	valor1@mail.com	valor1	valor1	123456	123456	-99.99	200.200.5.4	2018-01-22	2012/09/12

Tener en cuenta...

La tabla está compuesta por 14 columnas y se numera iniciando en (0) hasta (13).

Tiene dos filas y se numera iniciando en (0) hasta (13).

El campo Multiple Select, cuenta con dos columnas (MultipleS1 y MultipleS2).



Paso 2, Implementar la definition, recuerden que ya habíamos implementado el método “**PopupValidationDefinition**” en la clase “**diligencio_Formulario_Popup_Validation**”, por lo que ahora vamos a agregar el código necesario para que reciba por parámetros la tabla que se adjuntó en la **feature**.

Actualmente se encuentra así:

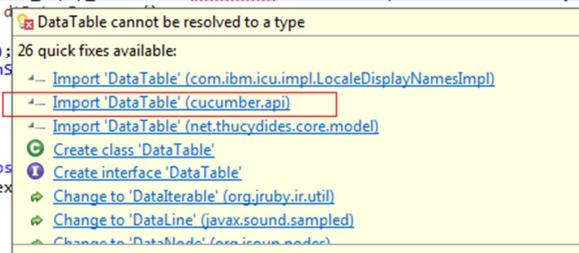
```
@When("^Diligencio Formulario Popup Validation$")
public void diligencio_Formulario_Popup_Validation() {
}
```

Paso 2.1, agregar al método el parámetro que va a recibir la información de la tabla, de tipo “**DataTable**” seguido del nombre que le queramos dar al objeto, para el ejemplo lo llamaremos **dtDatosForm**.

```
@When("^Diligencio Formulario Popup Validation$")
public void diligencio_Formulario_Popup_Validation(DataTable dtDatosForm) {
}
```

Nota: cuando utilice el tipo de objeto **DataTable**, importe la que corresponde a cucumber.api

```
@When("^Diligencio Formulario Popup Validation$")
public void diligencio_Formulario_Popup_Validation(DataTable dtDatosForm) throws Throwable {
    List<List<String>> data = dtDatosForm.raw();
    for(int i=1; i<data.size(); i++) {
        colorlibFormValidationSteps.diligenciar_popup_datos(data, i);
    }
}
@Then("^Verifico ingreso exitoso$")
public void verifico_ingreso_exitoso() {
}
```



The tooltip shows 'DataTable cannot be resolved to a type' and lists 26 quick fixes. One fix, 'Import 'DataTable' (cucumber.api)', is highlighted with a red box.

Paso 2.2, Crear un objeto tipo **List** que recibirá el datatable del parámetro.

```
List<List<String>> data = dtDatosForm.raw();
```

De aquí en adelante para hacer referencia a los datos de la tabla usaremos el objeto “data”.

Nota: existen varias formas de leer los datos del **DataTable** recibido:

<pre>String str_required = data.get(1).get(0).trim(); String str_select = data.get(1).get(1).trim(); String str_url = data.get(1).get(2).trim();</pre>	Asignar los valores recibidos a variables, donde: data.get(fila).get(columna).trim();
<pre>String[] DataPrueba = new String [data.size()]; for(int i=1; i<data.size(); i++){ DataPrueba[0] = data.get(i).get(0).trim(); DataPrueba[1] = data.get(i).get(1).trim(); DataPrueba[2] = data.get(i).get(2).trim(); colorlibFormValidationSteps.diligenciar_popup_datos(DataPrueba); } for(int i=1; i<data.size(); i++){ colorlibFormValidationSteps.diligenciar_popup_datos_tabla(data, i); }</pre>	Crear una matriz y asignar a cada posición de la matriz los valores de cada fila y columna del datatable.
	Usar el objeto tipo List<> y enviarlo por parámetro al método steps.

En nuestro caso usaremos la forma 3.



Paso 2.3, Instanciemos la clase steps que contendrá los pasos de diligenciamiento del formulario, la cual aún no existe pero que vamos a llamar “colorlibFormValidationSteps”.

Nota se debe agregar al inicio de la clase y anteponerse la anotación @Steps.

```
@Steps
PopupValidationSteps popupValidationSteps;
@Steps
colorlibFormValidationSteps colorlibFormValidationSteps;
```

Paso 2.4, dentro del método vamos a agregar un ciclo que nos permita consumir un método de diligenciamiento del formulario de tal forma que podamos recibir varios registros en el **datatable**.

```
29@When("^Diligencio Formulario Popup Validation$")
30    public void diligencio_Formulario_Popup_Validation(DataTable dtDatosForm) {
31        List<List<String>> data = dtDatosForm.raw();
32
33        for(int i=1; i<data.size(); i++){
34            colorlibFormValidationSteps.diligenciar_popup_datos_tabla(data, i);
35        }
}
```

En el código anterior, notamos que estamos proponiendo crear la clase “colorlibFormValidationSteps” y el método “diligenciar_popup_datos_tabla(data, i)” el cual recibirá el parámetro **data** (datatable) y el parámetro “**i**” (fila de datos).

Como la clase aún no existe vemos que se encuentra subrayada, por lo que procedemos a crear la clase en el paquete “com.choucair.formacion.steps”.

Nota: si la propiedad **raw**, se encuentra subrayada es por que no se importó correctamente la librería **cucumber.api.DataTable**.



Paso 3, implementar la clase steps, una vez creada la clase “colorlibFormValidationSteps”, procedemos a crear el método invocado desde la **definition**.

```
public void diligenciar_popup_datos_tabla(List<List<String>> data, int id) {  
}
```

Observe los parámetros de entrada del método, deben ser del mismo tipo de cómo fueron enviados desde la clase **definition**.

Paso 3.1, instanciar la clase en donde se definirán los objetos, aunque aún no existe dicha clase, vamos a instanciarla para que el sistema nos proponga crearla, vamos a llamarla “**ColorlibFormValidationPage**”

Adicional, recuerde agregar la anotación @Step antes del método.

```
public class colorlibFormValidationSteps {  
    ColorlibFormValidationPage colorlibFormValidationPage;  
    @Step  
    public void diligenciar_popup_datos_tabla(List<List<String>> data, int id) {
```

Paso 3.2, crear la clase para definir los objetos en el paquete “com.choucair.formacion.pageobjects”.



Paso 4, Definir los objetos de prueba en la clase "**ColorlibFormValidationPage**".

Nota: estos son algunas formas de identificar los objetos:

```
@FindBy(xpath="//*[@id='req']")
@FindBy(id="email1")
@FindBy(name="maxsize1")
@FindBy(cssSelector=".foo")
```

Recuerda cambiar la doble comilla por comilla simple en la identificación por xpath.

Paso 4.1, extender la clase a PageObject.

Paso 4.2, agregar la definición de cada objeto,

Nota: recuerde que puede apoyarse en **chrome** dando clic derecho sobre el objeto y luego la opción **inspeccionar**.

Una vez tengamos definidos todos los campos a utilizar tendríamos algo así:

```
ColorlibFo... ColorlibForm... PopupValida... PopupValida...
src/test/java com.choucair.formacion.pageobjects ColorlibFormVal
1 package com.choucair.formacion.pageobjects;
2
3 import net.serenitybdd.core.annotations.findby.FindBy;
4 import net.serenitybdd.core.pages.PageObject;
5 import net.serenitybdd.core.pages.WebElementFacade;
6
7 public class ColorlibFormValidationPage extends PageObject{
8
9 //Campo Required
10 @FindBy(xpath="//*[@id='req']")
11     public WebElementFacade txtRequired;
12 //Campo Seleccion de deporte
13 @FindBy(xpath="//*[@id='sport']")
14     public WebElementFacade cmbSport1;
15 //Campo Url
16 @FindBy(xpath="//*[@id='url1']")
17     public WebElementFacade txtUrl;
18 //Campo Email
19 @FindBy(id="email1")
20     public WebElementFacade txtEmail1;
21 //Campo Password
22 @FindBy(id="pass1")
23     public WebElementFacade txtPass1;
24 //Campo Password
25 @FindBy(id="pass2")
26     public WebElementFacade txtPass2;
27 //Campo Minsize
28 @FindBy(id="minsize1")
29     public WebElementFacade txtMinsize1;
30 //Campo Maxsize
31 @FindBy(name="maxsize1")
32     public WebElementFacade txtMaxsize;
33 //Campo Number
34 @FindBy(id="number2")
35     public WebElementFacade txtNumber;
36 //Campo IP
37 @FindBy(id="ip")
38     public WebElementFacade txtIp;
39 //Campo Date
40 @FindBy(id="date3")
41     public WebElementFacade txtDate;
42 //Campo Date Earlier
43 @FindBy(id="past")
44     public WebElementFacade txtDateEarlier;
45 }
```

Paso 4.3, crear los métodos para interactuar con cada objeto.

Para interactuar con cada objeto es necesario realizar tres acciones sobre el mismo, así que vamos a crear un método para la interacción con cada uno de los objetos, ejemplo:

```
public void Required(String datoPrueba) {
    txtRequired.click();
    txtRequired.clear();
    txtRequired.sendKeys(datoPrueba);
}
```

De esta forma, cuando vayamos a usar el objeto en los steps, sólo debemos llamar al método en este caso Required y enviarle el valor por parámetro.

Nota: esta es la forma de interactuar con un combobox

```
public void Select_Sport(String datoPrueba) {
    cmbSport1.click();
    cmbSport1.selectByVisibleText(datoPrueba);
}
```



En el caso del objeto Multiple Select, debemos crear el objeto una sola vez.

```
//Campo Seleccion multiple de deporte 2
@FindBy(xpath="//*[@id='sport2']")
public WebElementFacade cmbSport2;
```

Y creamos el siguiente método para la selección de los valores:

```
public void Multiple_Select(String datoPrueba) {
    cmbSport2.selectByVisibleText(datoPrueba);
}
```

Recuerde realizar la identificación del botón "Validate".

```
//Boton Validate
@FindBy(xpath ="//*[@id='popup-validation']/div[14]/input")
public WebElementFacade btnValidate;
```

Y del método que dará clic al objeto.

```
public void validate(){
    btnValidate.click();
}
```

Al finalizar la clase "*ColorlibFormValidationPage*", quedaría algo así:

(esta imagen es de referencia)

<pre>1 package com.choucair.formacion.pageobjects; 2 3 import net.serenitybdd.core.annotations.findby.FindBy; 4 5 6 public class ColorlibFormValidationPage { 7 //Campo Required 8 @FindBy(xpath="//*[@id='req']") 9 public WebElementFacade txtRequired; 10 //Campo Seleccion de deporte 1 11 @FindBy(xpath="//*[@id='sport']") 12 public WebElementFacade cmbSport1; 13 //Campo Url 14 @FindBy(xpath="//*[@id='url1']") 15 public WebElementFacade txtUrl; 16 //Campo Email 17 @FindBy(id="email1") 18 public WebElementFacade txtEmail1; 19 //Campo Password1 20 @FindBy(id="pass1") 21 public WebElementFacade txtPass1; 22 //Campo Password2 23 @FindBy(id="pass2") 24 public WebElementFacade txtPass2; 25 //Campo Minsize 26 @FindBy(id="minsize1") 27 public WebElementFacade txtMinsize1; 28 //Campo Maxsize 29 @FindBy(name="maxsize1") 30 public WebElementFacade txtMaxsize; 31 //Campo Number 32 @FindBy(id="number2") 33 public WebElementFacade txtNumber; 34 //Campo IP 35 @FindBy(id="ip") 36 public WebElementFacade txtIp; 37 //Campo Date 38 @FindBy(id="date3") 39 public WebElementFacade txtDate; 40 //Campo Date Earlier 41 @FindBy(id="past") 42 public WebElementFacade txtEarlier;</pre>	<pre>44 public void Required(String datoPrueba) { 45 txtRequired.click(); 46 txtRequired.clear(); 47 txtRequired.sendKeys(datoPrueba); 48 } 49 public void Select_Sport(String datoPrueba) { 50 cmbSport1.click(); 51 cmbSport1.selectByVisibleText(datoPrueba); 52 } 53 public void url(String datoPrueba) { 54 txtUrl.click(); 55 txtUrl.clear(); 56 txtUrl.sendKeys(datoPrueba); 57 } 58 public void email(String datoPrueba) { 59 txtEmail1.click(); 60 txtEmail1.clear(); 61 txtEmail1.sendKeys(datoPrueba); 62 } 63 public void password(String datoPrueba) { 64 txtPass1.click(); 65 txtPass1.clear(); 66 txtPass1.sendKeys(datoPrueba); 67 } 68 public void confirm_password(String datoPrueba) { 69 txtPass2.click(); 70 txtPass2.clear(); 71 txtPass2.sendKeys(datoPrueba); 72 } 73 public void minimun_field_size(String datoPrueba) { 74 txtMinsize1.click(); 75 txtMinsize1.clear(); 76 txtMinsize1.sendKeys(datoPrueba); 77 } 78 public void maximun_field_size(String datoPrueba) { 79 txtMaxsize.click(); 80 txtMaxsize.clear(); 81 txtMaxsize.sendKeys(datoPrueba); 82 } 83 public void number(String datoPrueba) { 84 txtNumber.click(); 85 txtNumber.clear(); 86 txtNumber.sendKeys(datoPrueba); 87 } 88 public void ip(String datoPrueba) {</pre>
--	--



Paso 5, Implementar los steps, como ya tenemos los objetos definidos es hora de regresar a los steps y agregar los pasos necesarios para diligenciar el formulario PopupValidation.

Recuerda que lo habíamos dejado inicializado así:

```

1 package com.choucair.formacion.steps;
2
3 import java.util.List;
4
5 public class colorlibFormValidationSteps {
6
7     ColorlibFormValidationPage colorlibFormValidationPage;
8
9     public void diligenciar_popup_datos_tabla(List<List<String>> data, int id) {
10
11     }
12
13
14 }
15
16

```

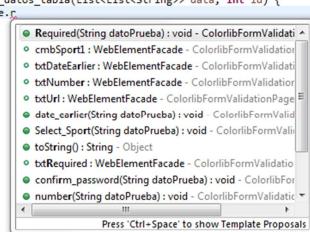
Comencemos a agregar los pasos, es muy simple:

Escriba `colorlibFormValidationPage`.y el sistema desplegará la lista de objetos y métodos definidos para ser usados.

```

public void diligenciar_popup_datos_tabla(List<List<String>> data, int id) {
    colorlibFormValidationPage.
    }

```



Por ejemplo para hacer referencia al diligenciamiento del campo “**Required**” haríamos referencia a la clase.método(valor):

```
colorlibFormValidationPage.Required(data.get(id).get(0).trim());
```

El valor corresponde a la posición del dato en el vector, es decir:

Get(Id) = corresponde a la fila enviada por parámetro desde la definición.

Get(#) = corresponde a la posición del dato en el vector por columnas, iniciando en cero.

En el caso del campo “Multiple Select”, recuerden que se definió el objeto una sola vez y un solo método, por lo que lo vamos a invocar dos veces enviéndole los valores de las posiciones 2 y 3 de la tabla:

```
colorlibFormValidationPage.Multiple_Select(data.get(id).get(2).trim());
colorlibFormValidationPage.Multiple_Select(data.get(id).get(3).trim());
```

Al final tendríamos algo así en los steps:

```

public class colorlibFormValidationSteps {
    ColorlibFormValidationPage colorlibFormValidationPage;

    @Step
    public void diligenciar_popup_datos_tabla(List<List<String>> data, int id) {
        colorlibFormValidationPage.Required(data.get(id).get(0).trim());
        colorlibFormValidationPage.Select_Sport(data.get(id).get(1).trim());
        colorlibFormValidationPage.Multiple_Select(data.get(id).get(2).trim());
        colorlibFormValidationPage.Multiple_Select(data.get(id).get(3).trim());
        colorlibFormValidationPage.url(data.get(id).get(4).trim());
        colorlibFormValidationPage.email(data.get(id).get(5).trim());
        colorlibFormValidationPage.password(data.get(id).get(6).trim());
        colorlibFormValidationPage.confirm_password(data.get(id).get(7).trim());
        colorlibFormValidationPage.minimum_field_size(data.get(id).get(8).trim());
        colorlibFormValidationPage.maximum_field_size(data.get(id).get(9).trim());
        colorlibFormValidationPage.number(data.get(id).get(10).trim());
        colorlibFormValidationPage.ip(data.get(id).get(11).trim());
        colorlibFormValidationPage.date(data.get(id).get(12).trim());
        colorlibFormValidationPage.date_earlier(data.get(id).get(13).trim());
        colorlibFormValidationPage.validate();
    }
}

```



Sólo nos falta implementar el paso o línea gherkin “Verifico ingreso exitoso”, vamos a hacerlo más rápido y resumido esta vez. Vamos a utilizar las clases ya existentes que usamos para diligenciar el formulario de validation.

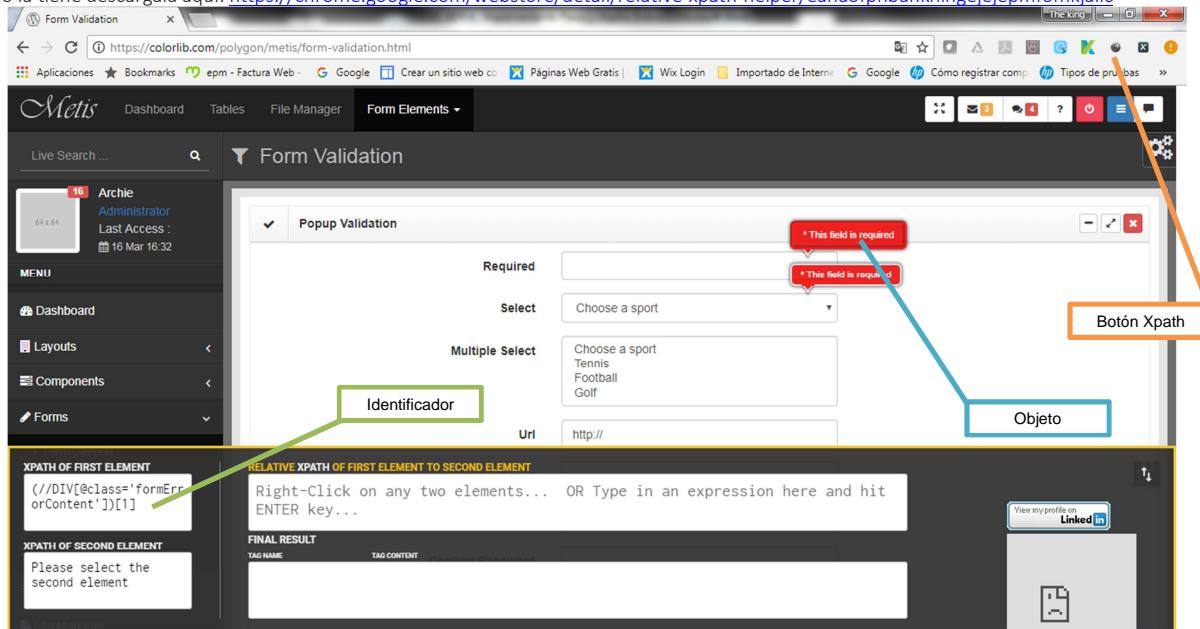
Paso 1, Crear el objeto en la clase “**ColorlibFormValidationPage**”.

Analicemos el patrón.

- La única forma de saber si el formulario se diligenció correctamente es porque no se presentó ningún globo informativo de validación de errores, por lo tanto debemos mapear el objeto “globo informativo”
- Para saber si el formulario se diligenció con errores con una casuística en particular, es porque se presentó un globo informativo con un texto en particular.
- El objeto a mapear pertenece a la misma clase por lo que tiene las mismas propiedades, es por esto que debemos utilizar un xpath relativo que nos permita identificar siempre el primer globo que se presente.

Apoyémonos en chrome, agregando la extensión “**Relative Xpath Helper**”, veamos:

Si no la tiene descárgala aquí: <https://chrome.google.com/webstore/detail/relative-xpath-helper/eanaofphbanknlngejejepmfmokjaiic>



- ✓ Click al botón “**Relative Xpath Helper**”, se activará la ventana de inspección.
- ✓ Clic derecho sobre el primer globo informativo.
- ✓ Tomar el xpath de la casilla “xpath of first element”

Paso 1.1, agregar la definición del objeto, lo vamos a llamar “globoInformativo”.

```
// Globo Informativo
  @FindBy(xpath="//DIV[@class='formErrorContent'])[1]")
  public WebElementFacade globoInformativo;
```

Paso 1.2, Crear el método encargado de verificar que no exista el objeto “globoInformativo”

```
public void form_sin_errores() {
  assertThat(globoInformativo.isCurrentlyVisible(), is(false));
}
```



Lo anterior lo hacemos con el método **assertThat**:

Sintaxis:

assertThat(recibido, esperado)
 recibido = tipo booleano (true, false), usaremos *globoInformativo.isCurrentlyVisible()*
 esperado = tipo booleano (true, false), usaremos *is(false)*

Aprovechemos que estamos creando los métodos y vamos a crear el método alterno, es decir verificar que si se encuentre el objeto.

```
public void form_con_errores() {
    assertThat(globoInformativo.isCurrentlyVisible(), is(true));
}
```

Recuerde: Para usar los comandos “*assertThat*” e “*is*”, agreguemos los siguientes import al inicio de la clase.

```
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.*;
```

Paso 2, Agregar los pasos a la clase “*colorlibFormValidationSteps*”.

```
@Step
public void verificar_ingreso_datos_formulario_exitoso() {
    colorlibFormValidationPage.form_sin_errores();
}
@Step
public void verificar_ingreso_datos_formulario_con_errores() {
    colorlibFormValidationPage.form_con_errores();
}
```

Paso 3, Implementar los pasos con la **definition** “*PopupValidationDefinition*”

```
@Then("^Verifico ingreso exitoso$")
public void verifco_ingreso_exitoso() {
    colorlibFormValidationSteps.verificar_ingreso_datos_formulario_exitoso();
}
```

Paso 4, ejecutar y generar el reporte de evidencias

Steps	Screenshot	Outcome	Duration
Given Autentico en colorlib con usuario "demo" y clave "demo"		SUCCESS	10.74s
And Ingreso a la funcionalidad Forms Validation		SUCCESS	1.62s
When Diligencio Formulario Popup Validation		SUCCESS	6.64s
Then Verifico ingreso exitoso		SUCCESS	0.63s
Verificar ingreso datos formulario exitoso		SUCCESS	0.34s
			19.84s



Paso 5, verificar el caso alterno.

Hasta el momento, nuestra automatización ingresa en el formulario los datos que hemos preparado para el caso exitoso, y la forma de identificar si el caso fue exitoso es porque no se presenta ningún globo informativo con la validación.

Como vimos en la evidencia el caso quedó marcado como exitoso.

Hagamos una pequeña prueba, en la que modificaremos uno de los datos de entrada de forma intencional, con el fin de hacer que se presente una validación en el formulario, en ese caso, la ejecución debería finalizar con error.

Paso 1, ingrese a la feature y modifique uno de los valores que se envían en el datatable.

```
When Diligencio Formulario Popup Validation
| Required | Select | MultipleS1| MultipleS2|Url           |Email           |Password1 |Password2 |
| Valor1   | Golf    | Tennis   | Golf      | http://www.valor1.com | valor1@mail.com | valor1    | valor1    |
```

Para el ejemplo vamos a modificar el valor del campo Url, quitando el prefijo http://.

```
When Diligencio Formulario Popup Validation
| Required | Select | MultipleS1| MultipleS2|Url           |Email           |Password1 |Password2 |Mi
| Valor1   | Golf    | Tennis   | Golf      | www.valor1.com | valor1@mail.com | valor1    | 123456   |
```

Ejecutemos nuevamente el caso, generamos evidencia y tendríamos el siguiente resultado:

Dar clic sobre el test.

Test Results: All Tests

2 test scenarios (3 tests in all, including 2 rows of test data)
| 1 pending | 1 unsuccessful (1 failed , 0 errors) | ↴

Test Type	Total	Pass	Fail	Pending	Ignores/Skip
Automated	3	0 (0%)	1 (33%)	2 (67%)	0 (0%)
Manual	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Total	3	0 (0%)	1 (33%)	2 (67%)	0 (0%)
Total Duration 19 seconds					

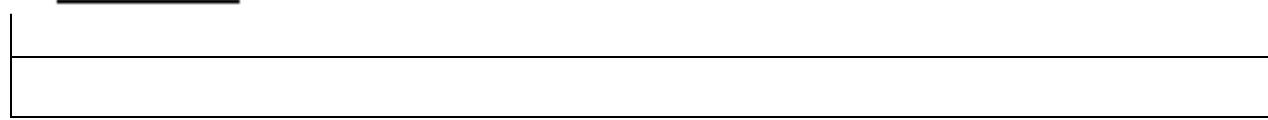
Related Tags

Tags	% Passed	Test count
Capabilities	0%	3
Features	0%	3
Formulario Popup Validation	0%	3
Tags	0%	3
Casoexitoso	0%	1
Regresion	0%	3
Tag2	0%	2
Themes	0%	3
Colorlib	0%	3

Tests

Entries	Search:
Tests	8
Diligenciamiento exitoso del formulario Popup Validation	15:45:41 19.26
Title of your scenario outline (2 examples)	8 17:37:23 0.02





Como podemos ver, el reporte nos muestra el error en la verificación.

The screenshot shows a test report for a 'Formulario Popup Validation' test. The 'Overall Test Results' section indicates a failure. The 'Diligenciamiento Exitoso Del Formulario Popup Validation' step is marked as 'FAILURE' with a red 'X' icon. The step details show a screenshot of the browser with a validation message: 'no se presenta ningún mensaje de validación.' Below this, a table lists the steps, their screenshots, outcomes, and durations:

Steps	Screenshot	Outcome	Duration
Given Autentico en colorlib con usuario "demo" y clave "demo"		SUCCESS	10.01s
And Ingreso a la funcionalidad Forms Validation		SUCCESS	1.74s
When Diligencio Formulario Popup Validation		SUCCESS	6.46s
Then Verifico ingreso exitoso		FAILURE	0.86s
Verificar ingreso datos formulario exitoso		FAILURE	0.85s

At the bottom of the report, a footer states 'Serenity BDD version 1.51' and shows a zoom level of '75%'.

Organicemos nuevamente el dato en la feature y guardar.

