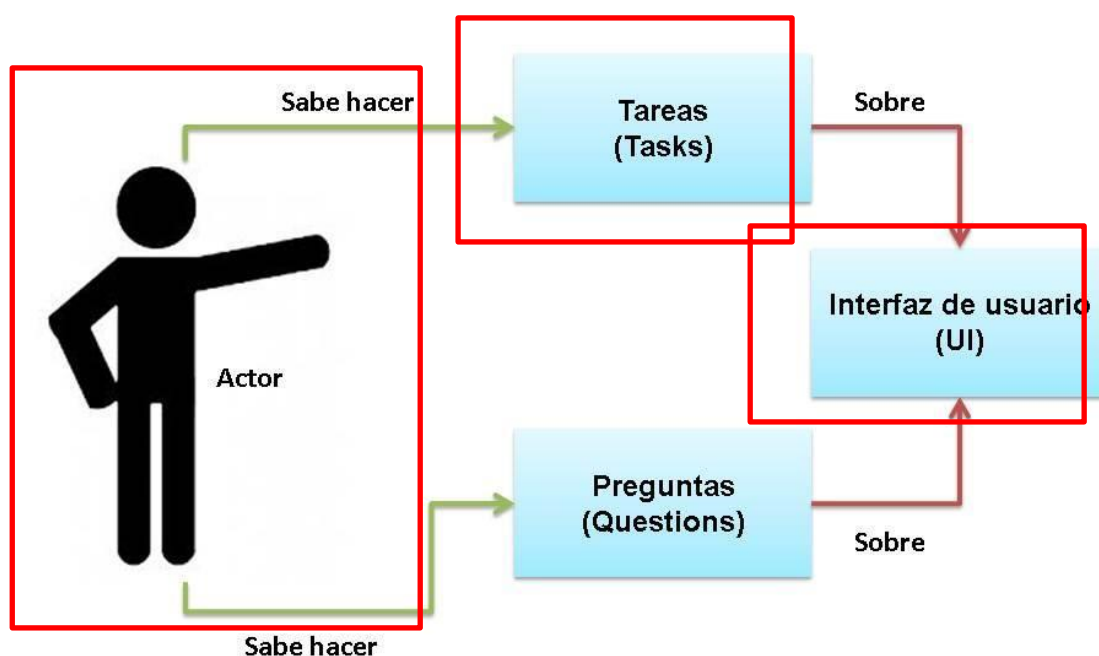


5 SERENITY BDD + SCREENPLAY con CUCUMBER

¡Bienvenidos a nuestra guía 5 del patrón Screenplay! En esta guía continuaremos con el aprendizaje de la implementación de tareas.



Aprenderemos cómo identificar los objetos en nuestro page y crearemos una nueva tarea.

¡Vamos a aprender!





Ahora, identificaremos los objetos de nuestra página web. Cuando damos un breve vistazo, podemos observar que son los siguientes.

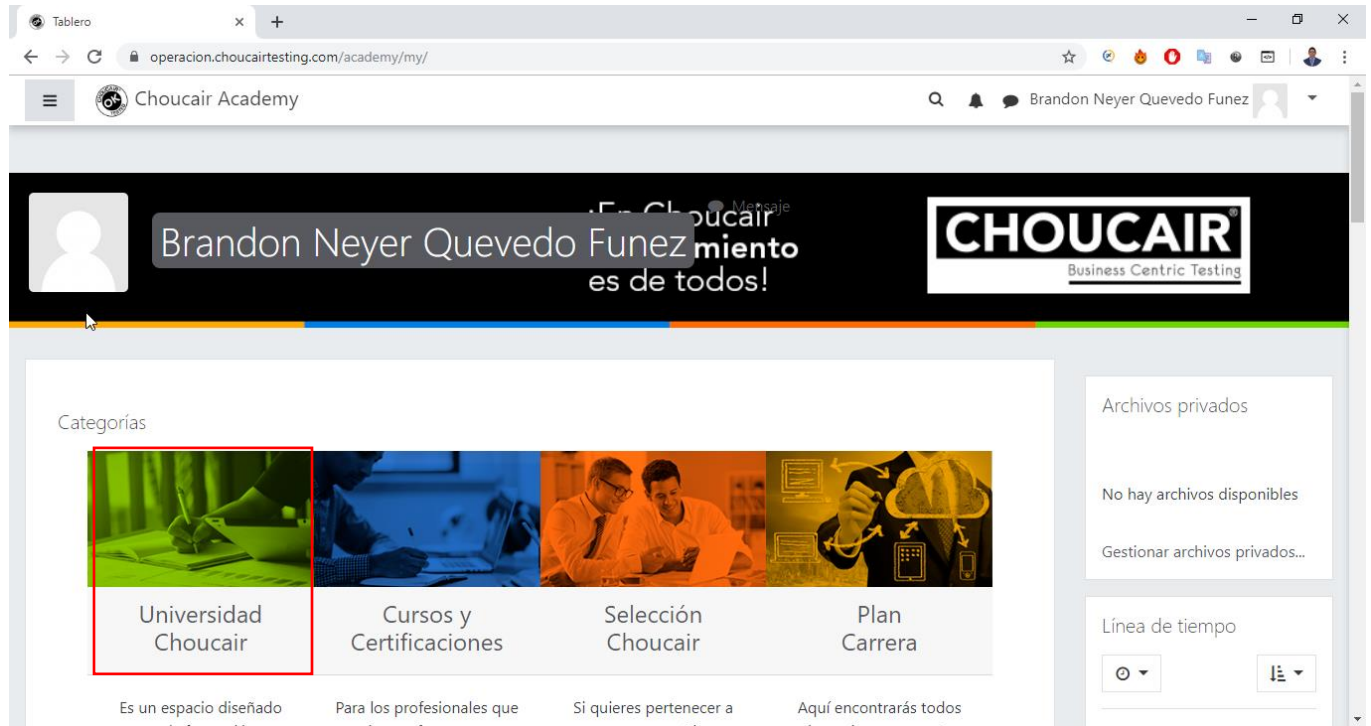


Se despliega una pantalla para el inicio de sesión. Donde mapearemos Nombre de usuario y Contraseña y el botón Acceder.

A login form titled "Iniciar Sesión" with a close button (X). It features the "CHOUCAIR OK TESTED" logo. Below the logo, there are two input fields: "Nombre de usuario" and "Contraseña", both highlighted with red rectangular boxes. Below these fields is a button labeled "Acceder", also highlighted with a red rectangular box. At the bottom of the form, there is a link that says "¿Has olvidado tu contraseña?".

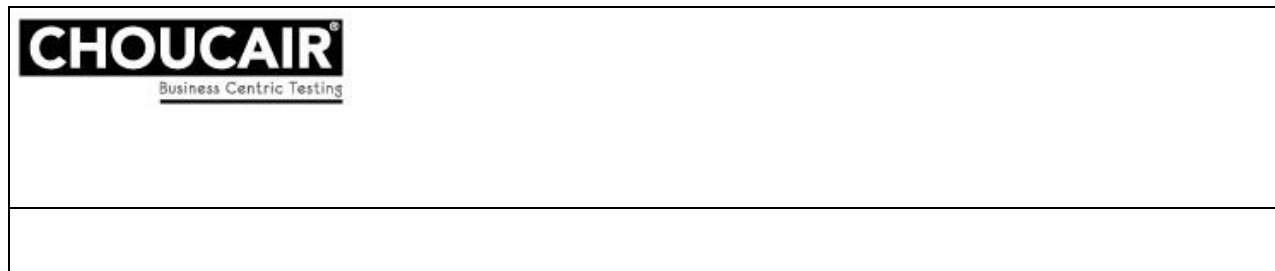


Damos clic en el link Universidad Choucair.



Buscamos el curso “Recursos Automatización Bancolombia” y damos clic en “ir”.

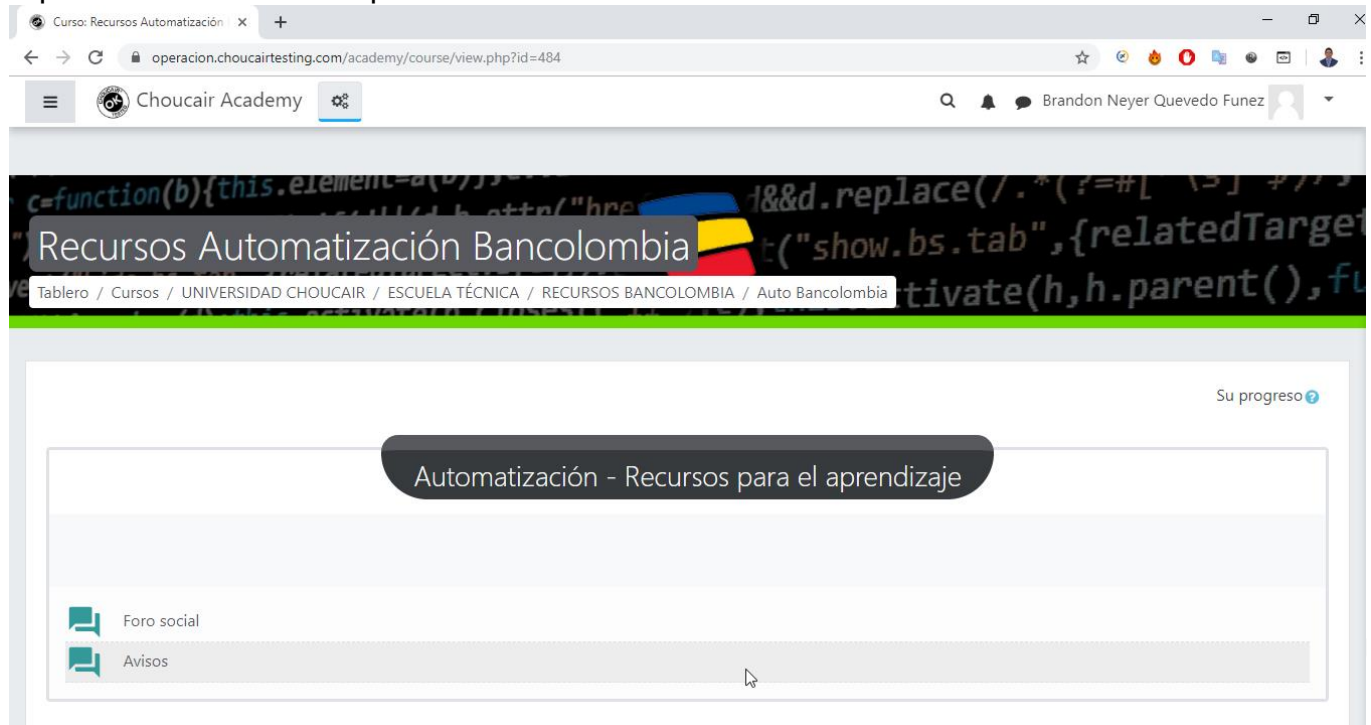




Damos clic en el resultado de la búsqueda.



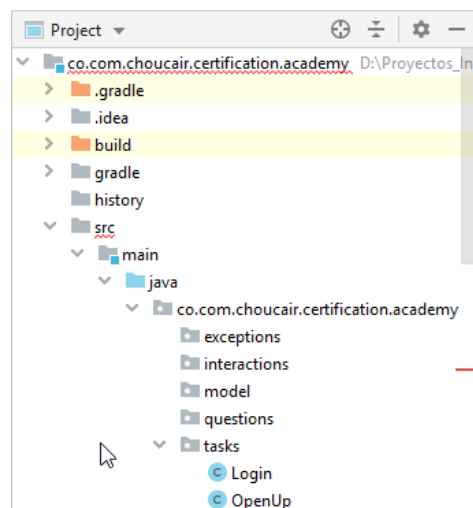
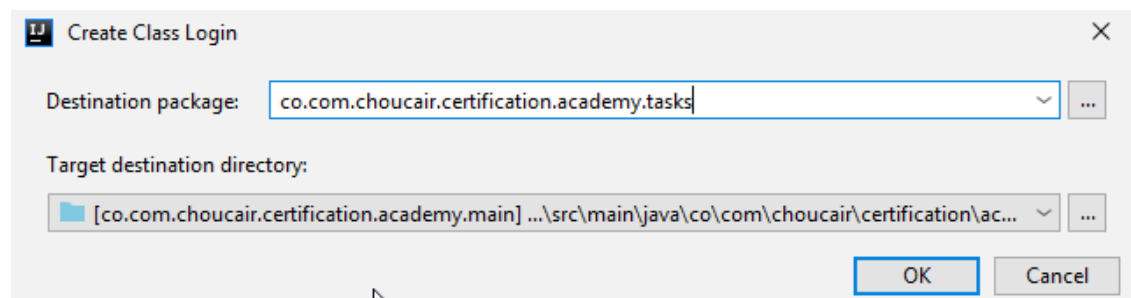
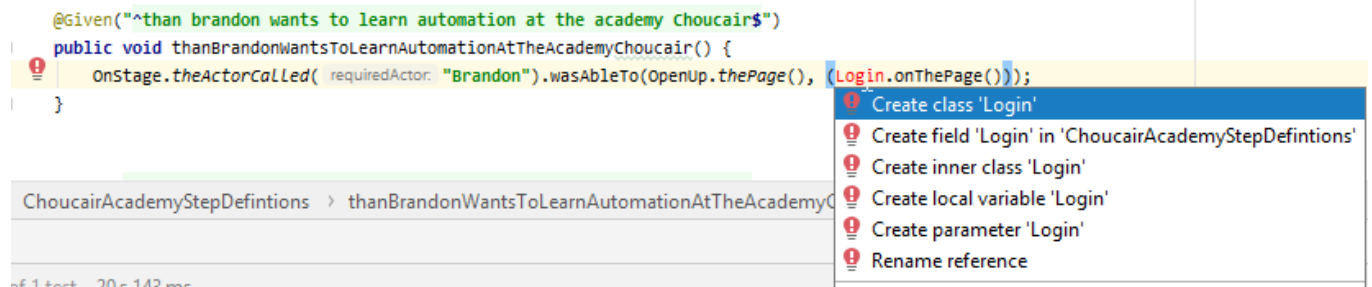
Y por último verificamos que el título si sea “Recursos Automatización Bancolombia”.



Debido a que necesitamos iniciar sesión en la página debemos crear otra precondition en el Given. Donde crearemos la Clase “Login” y el método “onThePage()”.

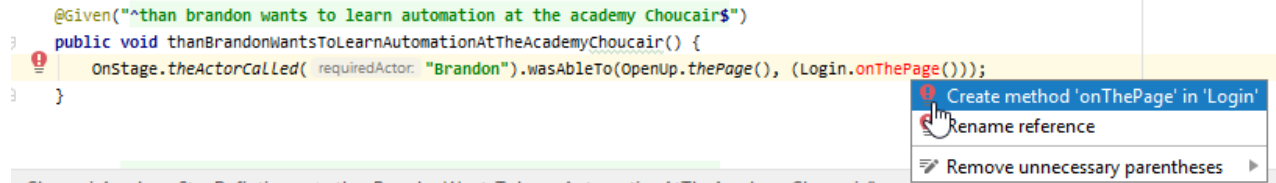
```
@Given("^than brandon wants to learn automation at the academy choucair$")
public void thanBrandonWantsToLearnAutomationAtTheAcademyChoucair() {
    OnStage.theActorCalled( requiredActor: "Brandon").wasAbleTo(OpenUp.thePage(), (Login.onThePage()));
}
```

Continuamos creando la clase “**Login**”, esta clase se alojará en el paquete “co.com.choucair.certification.academy.tasks”



Y el método “**onThePage()**” será un método estático de esta misma clase.

```
@Given("^than brandon wants to learn automation at the academy Choucair$")
public void thanBrandonWantsToLearnAutomationAtTheAcademyChoucair() {
    OnStage.theActorCalled( requiredActor: "Brandon").wasAbleTo(OpenUp.thePage(), (Login.onThePage()));
}
```



Después de crear la clase y método, debemos realizar el “*implements Task*”, agregar los métodos no implementados, cambiar el tipo “*Performable*” por el nombre de la clase y añadir el “*instrumented*” (**Según lo aprendido en la guía 2**), nuestra clase debe lucir de la siguiente forma:

```
1 package co.com.choucair.certification.academy.tasks;
2
3 import net.serenitybdd.screenplay.Actor;
4 import net.serenitybdd.screenplay.Task;
5 import net.serenitybdd.screenplay.Tasks;
6
7 public class Login implements Task {
8     public static Login onThePage() {
9         return Tasks.instrumented(Login.class);
10    }
11
12    @Override
13    public <T extends Actor> void performAs(T actor) {
14    }
15
16 }
```



¡Identifiquemos todos los objetos!

En la página inicial de Choucair Academy, lo primero que identificaremos será el botón que nos despliega el formulario para ingresar usuario y contraseña, lo haremos del mismo modo que lo sabemos hacer hasta el día de hoy, clic derecho sobre el elemento, inspeccionar. Sabiendo que buscaremos primero si tiene un **id**, un **name**, un **class**, y por último un buen **xpath** que nos reference el objeto.

En este caso, lo obtendremos por:

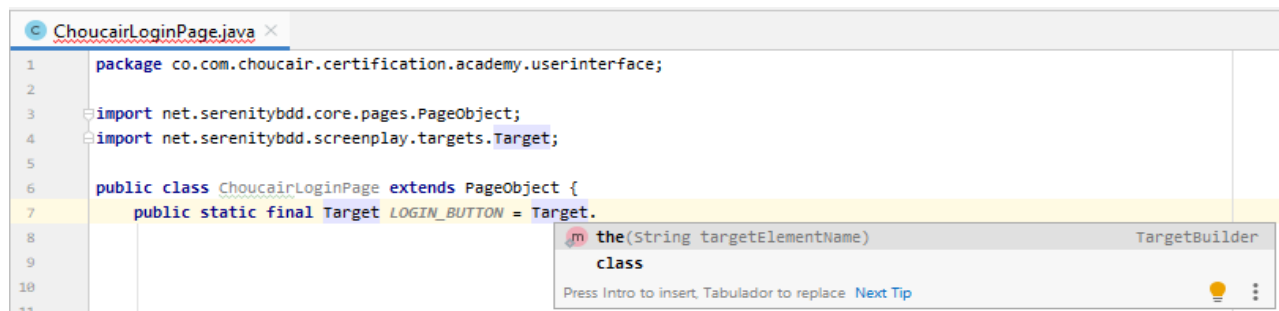
xpath = “//div[@class='d-none d-lg-block']/strong[contains(text(),'Ingresar')]”.

Iremos a nuestro paquete “co.com.choucair.certification.academy.userinterface” y crearemos la clase “ChoucairLoginPage”, y que extienda de “PageObject”.

Para la creación del objeto usaremos siempre las siguientes cuatro palabras más el nombre que deseemos darle a nuestro objeto:

public static final Target nombreDelObjeto

Para nuestra guía al botón de ingreso le llamaremos LOGIN_BUTTON. Para instanciar un objeto de tipo Target completaremos la línea con el siguiente código.



```
1 package co.com.choucair.certification.academy.userinterface;
2
3 import net.serenitybdd.core.pages.PageObject;
4 import net.serenitybdd.screenplay.targets.Target;
5
6 public class ChoucairLoginPage extends PageObject {
7     public static final Target LOGIN_BUTTON = Target.
8
9
10
11
```

TargetBuilder
the(String targetElementName)
class
Press Intro to insert, Tabulador to replace Next Tip

```
public static final Target LOGIN_BUTTON = Target.the("button that shows us the form to login").
```

Dentro del método the() escribiremos una breve descripción de lo que hace este objeto. Y completaremos la línea presionando punto y escogiendo el método “located”.

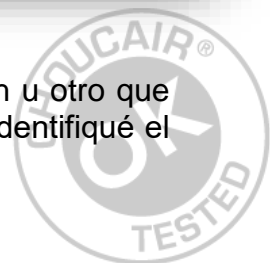



```
public class ChoucairLoginPage extends PageObject {  
    public static final Target LOGIN_BUTTON = Target.the( targetElementName: "button that shows us the form to login").  
    located(By locator) Target  
    locatedBy(String cssOrXPathSelector) Target  
    inIFrame(IFrame iframe) TargetBuild  
    equals(Object obj) boole  
    hashCode() i  
    toString() Stri  
    getClass() Class<? extends TargetBuilde  
    notify() vo  
    notifyAll() vo  
    wait() vo  
    wait(long timeout) vo  
    wait(long timeout, int nanos) vo  
}
```

Dentro del método escribiremos By y presionamos punto (.) y se abrirá otro menú....

```
public static final Target LOGIN_BUTTON = Target.the( targetElementName: "button that shows us the form to login").located(By.).  
    By.xpath(String xpathExpression) (org.openqa.selenium)  
    By.xpath(String xpathExpression) (net.thucydides.core.annotations.findb_  
    By.xpath(String xpathExpression) (net.serenitybdd.core.annotations.find_  
    By.id(String id) (org.openqa.selenium)  
    By.id(String id) (net.thucydides.core.annotations.findby)  
    By.id(String id) (net.serenitybdd.core.annotations.findby)  
    By.scLocator(String scLocator) (net.thucydides.core.annotations.findby)  
    By.scLocator(String scLocator) (net.serenitybdd.core.annotations.findby)  
    By.buttonText(String text) (net.thucydides.core.annotations.findby)  
    By.buttonText(String text) (net.serenitybdd.core.annotations.findby)  
    By.className(String className) (org.openqa.selenium)  
    By.className(String className) (net.thucydides.core.annotations.findby)  
    By.className(String className) (net.serenitybdd.core.annotations.findby)  
    By.cssSelector(String selector) (org.openqa.selenium)  
    By.cssSelector(String selector) (net.thucydides.core.annotations.findby)  
    By.cssSelector(String selector) (net.serenitybdd.core.annotations.findb_  
    By.jquery(String jquerySelector) (net.thucydides.core.annotations.findb_  
    By.jquery(String jquerySelector) (net.serenitybdd.core.annotations.find_  
    By.linkText(String linkText) (org.openqa.selenium)  
    By.linkText(String linkText) (net.thucydides.core.annotations.findby)  
Press Intro to insert, Tabulador to replace Next Tip
```

Y escogemos la opción que deseemos, ya sea id, name, className, xpath u otro que nos ayude a identificar el elemento. Por último, añadimos el “nombre” que identifiqué el objeto. En nuestro caso inspeccionamos el objeto con un xpath.



Nuestra línea de código queda de la siguiente forma:

```
public static final Target LOGIN_BUTTON = Target.the( targetElementName: "button that shows us the form to login")
    .located(By.xpath("//div[@class='d-none d-lg-block']//strong[contains(text(),'Ingresar')]"));
```

Ahora, debemos identificar todos los demás objetos que usaremos en este ejercicio. Puedes empezar a poner en práctica la identificación de objetos antes de continuar con la guía, si lo deseas. Para esta página identificaremos también los inputs y el botón acceder.

Recuerda puede variar la forma en que hayas identificado tus objetos con respecto a las del ejemplo, recuerda que puedes identificarlos por id, name, xpath, entre otros.

A continuación, en la imagen se muestra un ejemplo:

```
9      public static final Target LOGIN_BUTTON = Target.the( targetElementName: "button that shows us the form to login")
10      |      .located(By.xpath("//div[@class='d-none d-lg-block']//strong[contains(text(),'Ingresar')]"));
11      public static final Target INPUT_USER =Target.the( targetElementName: "where do we write the user")
12      |      .located(By.id("username"));
13      public static final Target INPUT_PASSWORD =Target.the( targetElementName: "where do we write the password")
14      |      .located(By.id("password"));
```

¡Vas bien tú puedes!



Una vez realizada la identificación de los objetos, tendremos una clase con la siguiente información:

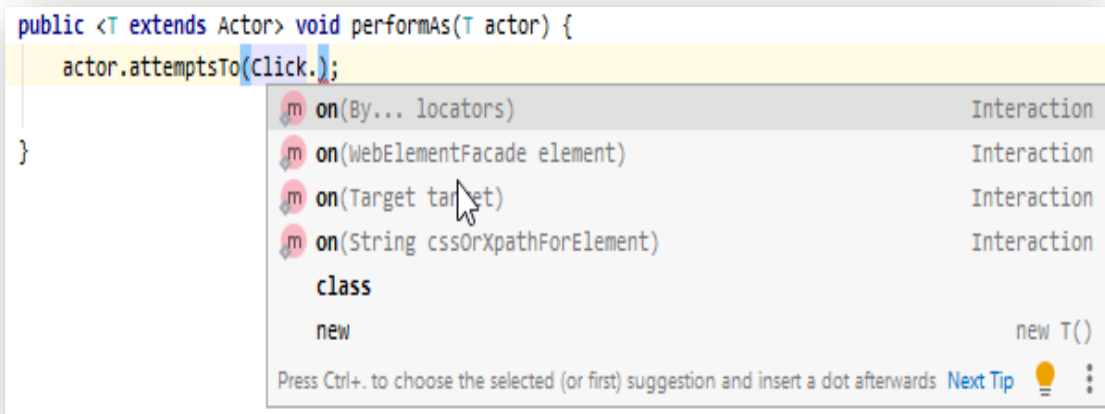
```
ChoucairLoginPage.java X
1 package co.com.choucair.certification.academy.userinterface;
2
3 import net.serenitybdd.core.pages.PageObject;
4 import net.serenitybdd.screenplay.targets.Target;
5 import org.openqa.selenium.By;
6
7 public class ChoucairLoginPage extends PageObject {
8
9     public static final Target LOGIN_BUTTON = Target.the( targetElementName: "button that shows us the form to login")
10         .located(By.xpath("//div[@class='d-none d-lg-block']/strong[contains(text(),'Ingresar')]"));
11     public static final Target INPUT_USER = Target.the( targetElementName: "where do we write the user")
12         .located(By.id("username"));
13     public static final Target INPUT_PASSWORD = Target.the( targetElementName: "where do we write the password")
14         .located(By.id("password"));
15     public static final Target ENTER_BUTTON = Target.the( targetElementName: "button to confirm login")
16         .located(By.xpath("//button[contains(@class,'btn btn-primary')]"));
17
18 }
19
```

Una vez terminemos con los objetos volveremos a nuestra clase Login (task) y efectuaremos todas las interacciones entre el actor y estos objetos.

```
Login.java X
1 package co.com.choucair.certification.academy.tasks;
2
3 import net.serenitybdd.screenplay.Actor;
4 import net.serenitybdd.screenplay.Task;
5 import net.serenitybdd.screenplay.Tasks;
6
7 public class Login implements Task {
8     public static Login onThePage() {
9         return Tasks.instrumented(Login.class);
10     }
11
12     @Override
13     public <T extends Actor> void performAs(T actor) {
14
15     }
16 }
```



Recuerda que todo nuestro código irá siempre en el método `performAs`. Iniciaremos escribiendo `actor.attemptsTo()` y haremos cada paso para realizar el logueo.



Escribiremos “Click.on” Y le pasaremos como parámetro la página que contiene al objeto deseado (`ChoucairLoginPage`) seguido de un punto, con el nombre del objeto que compete (`LOGIN_BUTTON`).

```
actor.attemptsTo(Click.on(ChoucairLoginPage.LOGIN_BUTTON));
```

Añadimos todas las instrucciones que se requieren para lograr iniciar sesión, separando por “,” cada instrucción:

```
actor.attemptsTo(Click.on(ChoucairLoginPage.LOGIN_BUTTON),  
    Enter.theValue("IngresaTuUsuario").into(ChoucairLoginPage.INPUT_USER),  
    Enter.theValue("IngresaTuContraseña*").into(ChoucairLoginPage.INPUT_PASSWORD),  
    Click.on(ChoucairLoginPage.ENTER_BUTTON)  
);
```

De esta manera interactuamos con objetos a los cuales se les puede realizar la acción de dar “Click”, para escribir en campos entonces usaremos el método “Enter”, y lo haremos como en la imagen anterior.



La clase Login nos quedaría así:

```
1 package co.com.choucair.certification.academy.tasks;
2
3 import co.com.choucair.certification.academy.userinterface.ChoucairLoginPage;
4 import net.serenitybdd.screenplay.Actor;
5 import net.serenitybdd.screenplay.Task;
6 import net.serenitybdd.screenplay.Tasks;
7 import net.serenitybdd.screenplay.actions.Click;
8 import net.serenitybdd.screenplay.actions.Enter;
9
10 public class Login implements Task {
11     public static Login onThePage() {
12         return Tasks.instrumented(Login.class);
13     }
14
15     @Override
16     public <T extends Actor> void performAs(T actor) {
17         actor.attemptsTo(Click.on(ChoucairLoginPage.LOGIN_BUTTON),
18             Enter.theValue("ingresarTuUsuario").into(ChoucairLoginPage.INPUT_USER),
19             Enter.theValue("ingresarTuContraseña").into(ChoucairLoginPage.INPUT_PASSWORD),
20             Click.on(ChoucairLoginPage.ENTER_BUTTON)
21         );
22     }
23 }
24
```

Login > onThePage()

Y nuestro StepDefintions en el Given se vería así:

```
@Given("^than brandon wants to learn automation at the academy Choucair$")
public void thanBrandonWantsToLearnAutomationAtTheAcademyChoucair() {
    OnStage.theActorCalled(requiredActor: "Brandon").wasAbleTo(OpenUp.thePage(), (Login.onThePage()));
}
```



Con el objetivo de dar continuidad a nuestra historia de usuario, desarrollaremos nuestra segunda actividad, la cual es:

When he search for the course on the choucair academy platform

Lo primero que haremos es ir a nuestra clase **“ChoucairAcademyStepDefintions”** y crear la línea que nos ejecutará la tarea, de la forma en que aprendimos en la guía anterior.

Al estar implementando código en el **“When”**, el método que usaremos será el `attemptTo()`. Incluyamos entonces la siguiente línea de código:

```
OnStage.theActorInTheSpotlight().attemptTo(Search.the(course));
```

Recuerda, **“Search”**, será una clase que irá en el paquete `“co.com.choucair.certification.academy.tasks”` y **“the()”** será un método estático de esta misma clase. Haremos una sutil modificación en nuestro método en la clase **“ChoucairAcademyStepDefintions”**, antes de pasar a crear las clases que correspondan.

Nuestro método ahora mismo luce de la siguiente forma:

```
@When("^he search for the course Recursos Automatización Bancolombia on the choucair academy platform$")
public void heSearchForTheCourseRecursosAutomatizaciónBancolombiaOnTheChoucairAcademyPlatform() {
```

Vamos a convertir la frase **“Recursos Automatización Bancolombia”** en una expresión regular, para hacer nuestro método dinámico, por lo tanto, haciendo los cambios, obtenemos el siguiente resultado:

```
@When("^he search for the course (.*) on the choucair academy platform$")
public void heSearchForTheCourseRecursosAutomatizaciónBancolombiaOnTheChoucairAcademyPlatform(String course) {
```

Una vez hecho estos cambios, creamos nuestra clase **“Search”** y nuestro método **“the(course)”**.



```
@When("^he search for the course (.*) on the choucair academy platform$")
public void heSearchForTheCourseRecursosAutomatizaciónBancolombiaOnTheChoucairAcademyPlatform(String course) {
    OnStage.theActorInTheSpotlight().attemptsTo(Search.the(course));
}

@Then("^he finds the course called resources Recursos$")
public void heFindsTheCourseCalledResourcesRecursos() {
}
```

Import class
Create class 'Search'
Create 'Search' in 'ChoucairAcademyStepDefintions'
Create inner class 'Search'
Create local variable 'Search'
Create parameter 'Search'
Rename reference

Creación del Método

```
@When("^he search for the course (.*) on the choucair academy platform$")
public void heSearchForTheCourseRecursosAutomatizaciónBancolombiaOnTheChoucairAcademyPlatform(String course) {
    OnStage.theActorInTheSpotlight().attemptsTo(Search.the(course));
}
```

Create method 'the' in 'Search'
Rename reference

Luego de crear clase y método, debemos implementar las tareas, agregar los métodos no implementados, cambiar el tipo “*Performable*” por el nombre de la clase y añadir el “*instrumented*” (**Según lo aprendido anteriormente**), nuestra clase debe lucir de la siguiente forma:

```
Search.java
1 package co.com.choucair.certification.academy.tasks;
2
3 import net.serenitybdd.screenplay.Actor;
4 import net.serenitybdd.screenplay.Task;
5 import net.serenitybdd.screenplay.Tasks;
6
7 public class Search implements Task {
8
9     public static Search the(String course) {
10         return Tasks.instrumented(Search.class);
11     }
12
13     @Override
14     public <T extends Actor> void performAs(T actor) {
15
16     }
17 }
18
```



Recuerda, que lo descrito anteriormente es un proceso repetitivo, que SIEMPRE será igual en la construcción de una tarea (Task).

Ahora procedemos a crear otra clase para mapear los elementos de la búsqueda, esta clase se llamará SearchCoursePage y pertenece al paquete “userinterface” y también extiende de PageObject.

```
SearchCoursePage.java ×
1 package co.com.choucair.certification.academy.userinterface;
2
3 import net.serenitybdd.core.pages.PageObject;
4
5 public class SearchCoursePage extends PageObject {
6 }
```

En esta clase se debemos identificar todos los demás objetos que usaremos para la búsqueda del curso.

**Que te parece si ahora lo intentas tu
antes de continuar con la siguiente pagina**



Felicitaciones vas muy bien, sigue así...

Nuestra clase debió quedar muy parecida a esta:

```
SearchCoursePage.java
1 package co.com.choucair.certification.academy.userinterface;
2
3 import net.serenitybdd.core.pages.PageObject;
4 import net.serenitybdd.screenplay.targets.Target;
5 import org.openqa.selenium.By;
6
7 public class SearchCoursePage extends PageObject {
8     public static final Target BUTTON_UC = Target.the( targetElementName: "Selecciona la universidad choucair")
9         .located(By.xpath("//div[@id='universidad']//strong"));
10    public static final Target INPUT_COURSE = Target.the( targetElementName: "Buscar el curso")
11        .located(By.id("coursesearchbox"));
12    public static final Target BUTTON_GO = Target.the( targetElementName: "Da click para buscar el curso")
13        .located(By.id("//button[@class='btn btn-secondary']"));
14    public static final Target SELECT_COURSE = Target.the( targetElementName: "Da click para buscar el curso")
15        .located(By.xpath("//h4[contains(text(),'Recursos Automatización Bancolombia')]"));
16
17 }
```

Una vez terminemos con los objetos volveremos a nuestra clase Search (task) y efectuaremos todas las interacciones entre el actor y estos objetos.

```
Search.java
1 package co.com.choucair.certification.academy.tasks;
2
3 import net.serenitybdd.screenplay.Actor;
4 import net.serenitybdd.screenplay.Task;
5 import net.serenitybdd.screenplay.Tasks;
6
7 public class Search implements Task {
8
9     public static Search the(String course) { return Tasks.instrumented(Search.class); }
10
11     @Override
12     public <T extends Actor> void performAs(T actor) {
13
14     }
15
16 }
17
18
```

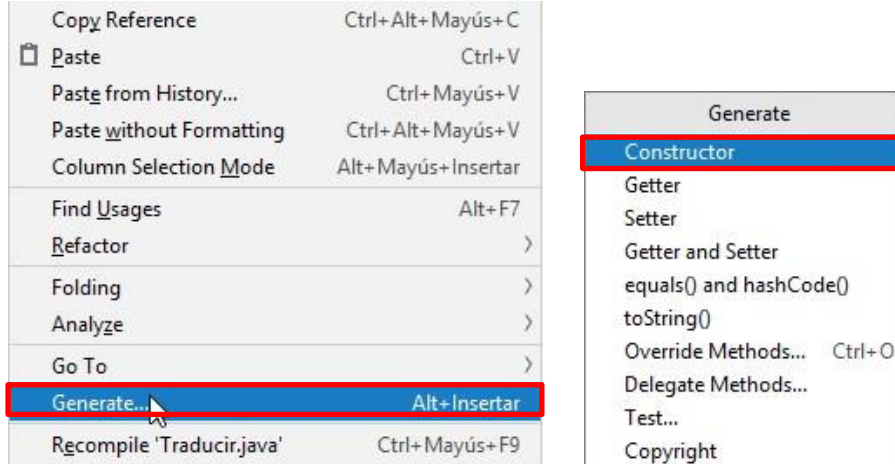


El parámetro “**course**” deberá ser declarado como un atributo de clase de tipo String y encapsulado de la siguiente forma:

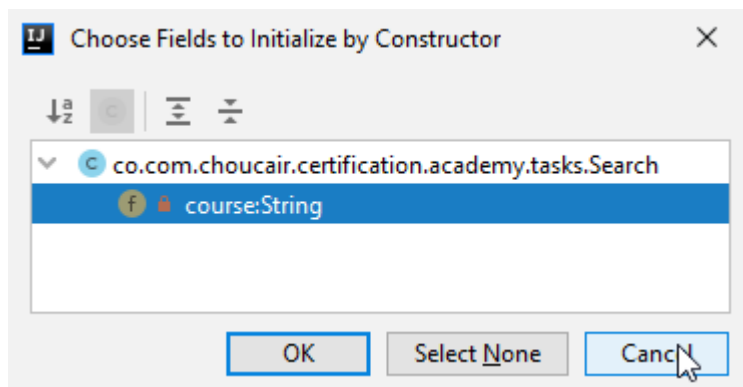
```
private String course;
```

Debido a que esta clase cuenta, con un atributo de tipo String llamado “course” tendremos la obligación de definirle un Constructor a nuestra clase.

Una manera sencilla de crear un constructor con los atributos definidos es la siguiente: Estando sobre la clase hacemos clic derecho y nos dirigimos donde indica la ruta, navegamos en el menú hasta la opción “Generate” y luego clic en “Constructor”



Seleccionamos el campo y damos clic en “OK”

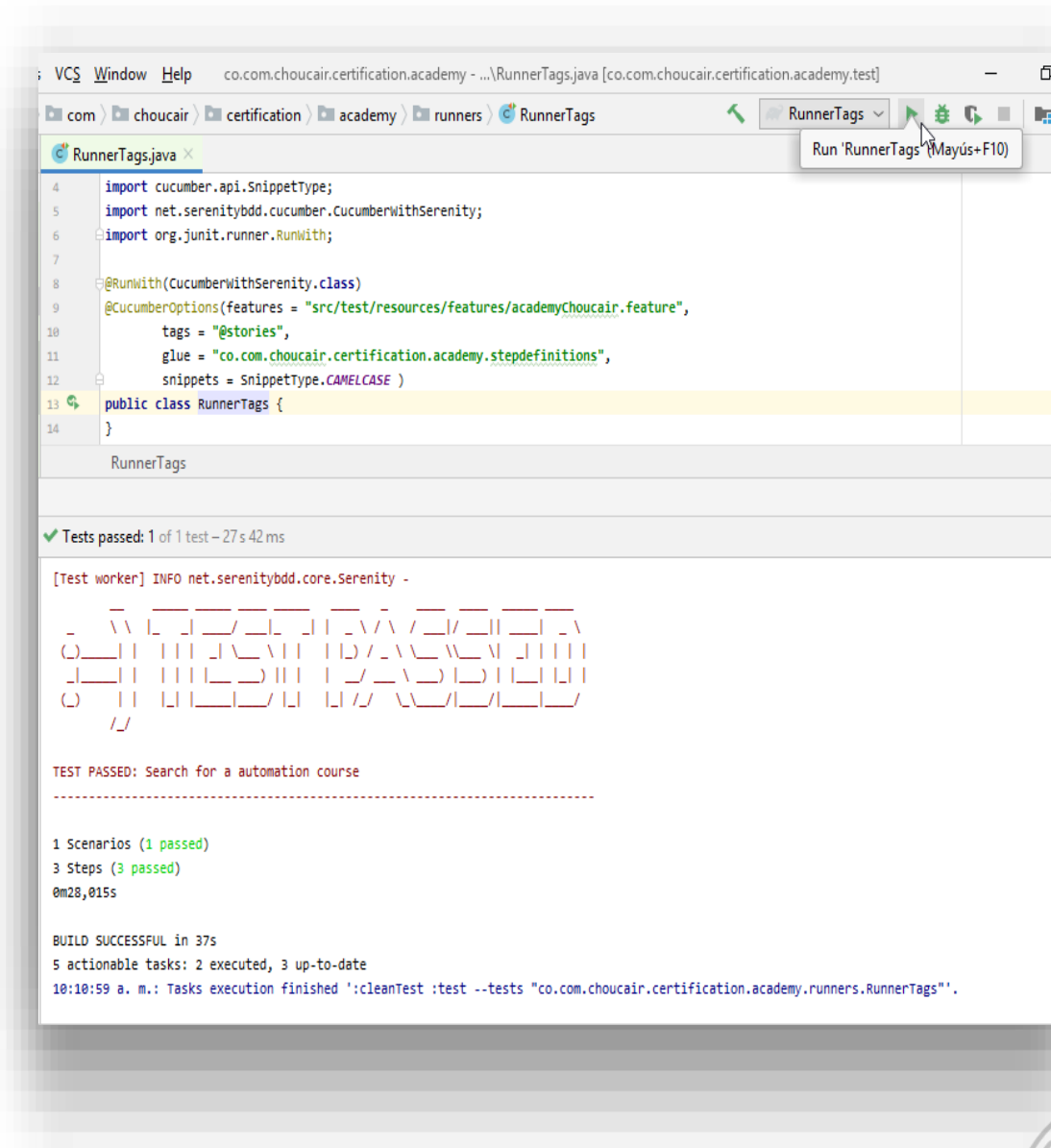


Finalmente, debido a que ahora tenemos un constructor y dicho constructor maneja unos parámetros, tendremos que modificar nuestro método “the()”, en el método instrumented, le añadiremos el parámetro “course”, obteniendo una clase como la siguiente:

```
1 package co.com.choucair.certification.academy.tasks;
2
3 import co.com.choucair.certification.academy.userinterface.SearchCoursePage;
4 import net.serenitybdd.screenplay.Actor;
5 import net.serenitybdd.screenplay.Task;
6 import net.serenitybdd.screenplay.Tasks;
7 import net.serenitybdd.screenplay.actions.Click;
8 import net.serenitybdd.screenplay.actions.Enter;
9
10 public class Search implements Task {
11     private String course;
12
13     @ public Search(String course) {
14         this.course = course;
15     }
16
17     public static Search the(String course) { return Tasks.instrumented(Search.class, course); }
18
19     @Override
20     public <T extends Actor> void performAs(T actor) {
21         actor.attemptsTo(Click.on(SearchCoursePage.BUTTON_UC),
22             Enter.theValue(course).into(SearchCoursePage.INPUT_COURSE),
23             Click.on(SearchCoursePage.BUTTON_GO),
24             Click.on(SearchCoursePage.SELECT_COURSE)
25         );
26     }
27 }
28
29 Search > Search()
```



Ahora procedemos a ejecutar nuestra prueba y podremos observar que ya hay una interacción con los objetos y el resultado exitoso de la misma. **TEST PASSED.**



¡NOS VEMOS EN LA PRÓXIMA GUÍA!

