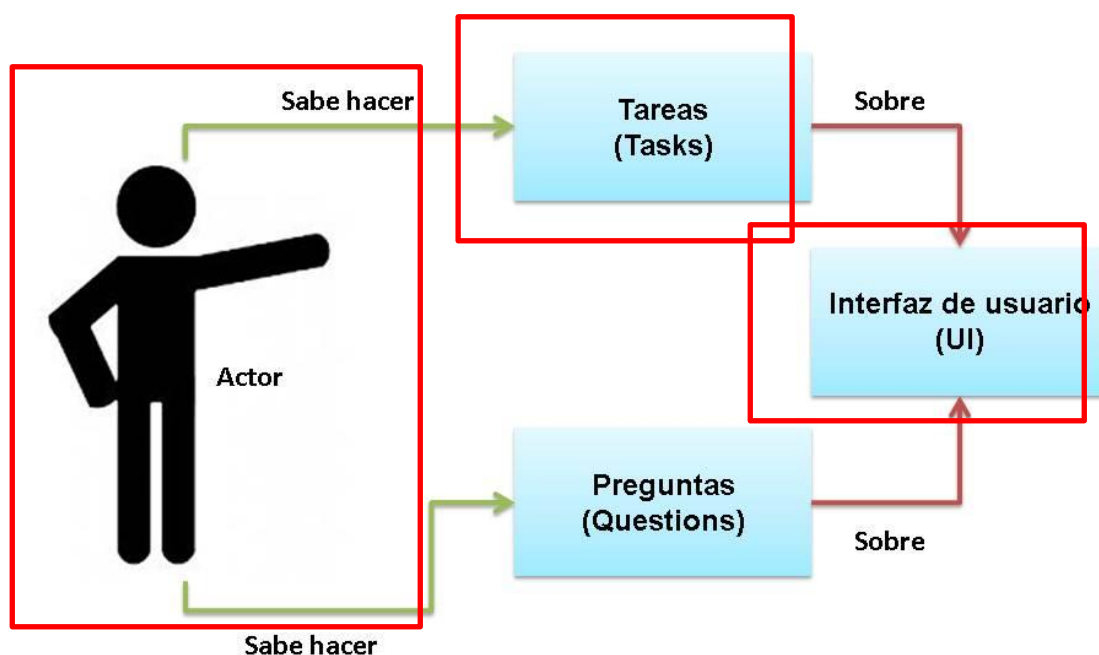


5

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!



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

When el traduce la palabra table del inglés al español

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

<u>Clase</u>	<u>Método static</u>
Accion .	Complemento de la acción ()

Ejemplos:

```
rafa.wasAbleTo(Ingresar.ALaPaginaDeGoogle());  
rafa.wasAbleTo(Abrir.PaginaInicialDeYoutube());  
rafa.attemptsTo(Buscar.SuCancionFavorita());  
rafa.attemptsTo(Loguearse.ConLasSiguietesCredenciales(usuario, clave));
```

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

```
rafa.attemptsTo(Traducir.DeInglesAEspanolLa(palabra));
```

Recuerda, **“Traducir”**, será una clase que irá en el paquete **“co.com.proyectobase.screenplay.tasks”** y **“DeInglesAEspanolLa()”** será un método estático de esta misma clase. Haremos una sutil modificación en nuestro método en la clase **“TraductorGoogleStepDefinition”**, antes de pasar a crear las clases que correspondan.

Nuestro método ahora mismo luce de la siguiente forma:

```
@When("^el traduce la palabra table del inglés al español$")  
public void elTraduceLaPalabraTableDelInglésAlEspañol() throws Exception {
```

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



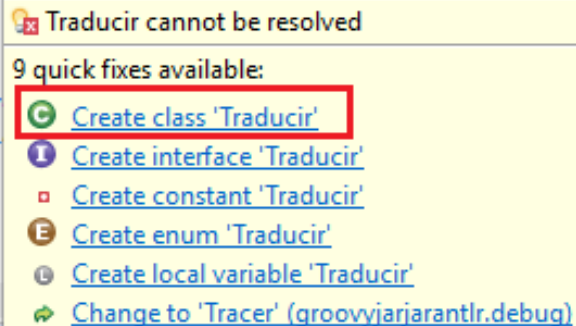
```
@When("^el traduce la palabra (.*) del inglés al español$")
public void elTraduceLaPalabraTableDelInglésAlEspañol(String palabra) {
```

Una vez hecho estos cambios, creamos nuestra clase Traducir y nuestro método “DeInglesAEspanolLa(palabra)”.

```
@When("^el traduce la palabra (.*) del inglés al español$")
public void elTraduceLaPalabraTableDelInglésAlEspañol(String palabra) {
    rafa.attemptsTo(Traducir.DeInglesAEspanol(palabra));
```



```
@Then("^el debería v
public void elDeberia
```

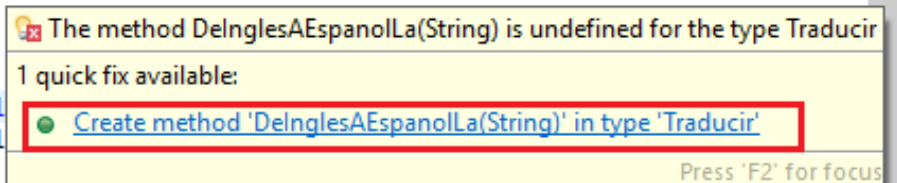


Creación del Método

```
@When("^el traduce la palabra (.*) del inglés al español$")
public void elTraduceLaPalabraTableDelInglésAlEspañol(String palabra) {
    rafa.attemptsTo(Traducir.DeInglesAEspanolLa(palabra));
```

```
}
```

```
@Then("^el debería ver la pal
public void elDeberiaVerLaPal
```



Después de crear 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:

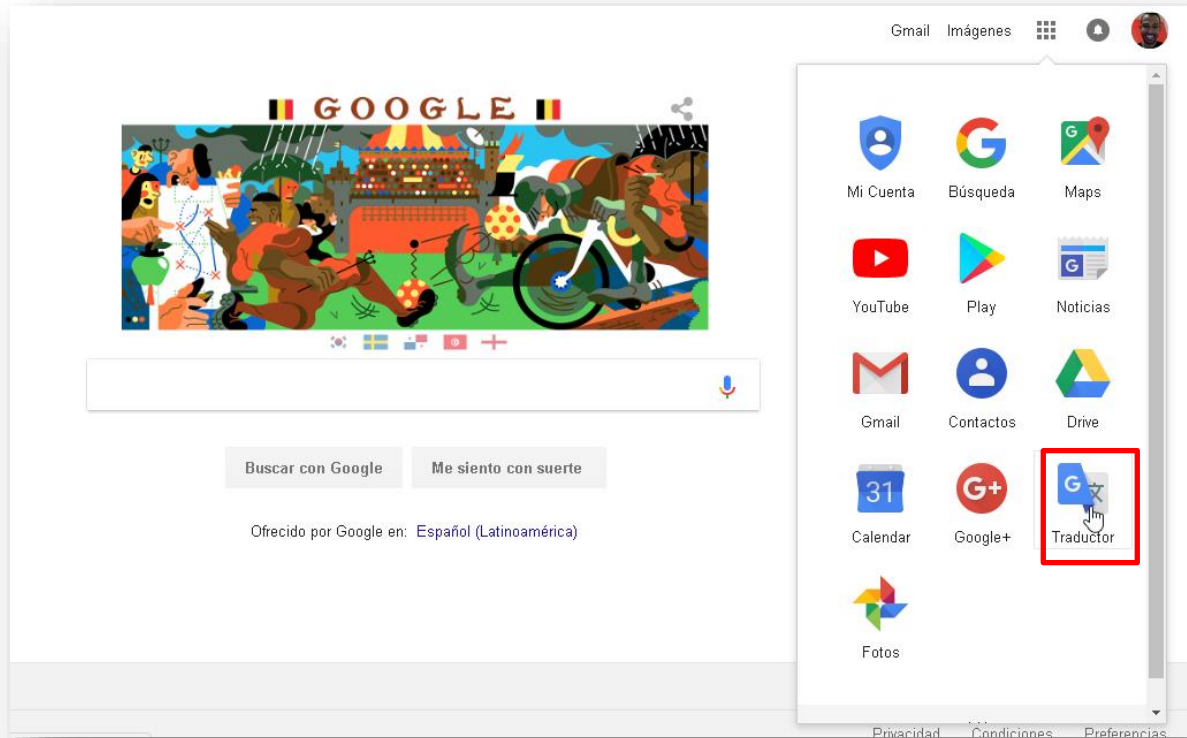
```
1 package co.com.proyectobase.screenplay.tasks;
2
3 import net.serenitybdd.screenplay.Actor;
4 import net.serenitybdd.screenplay.Task;
5 import net.serenitybdd.screenplay.Tasks;
6
7 public class Traducir implements Task{
8
9     @Override
10    public <T extends Actor> void performAs(T actor) {
11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

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

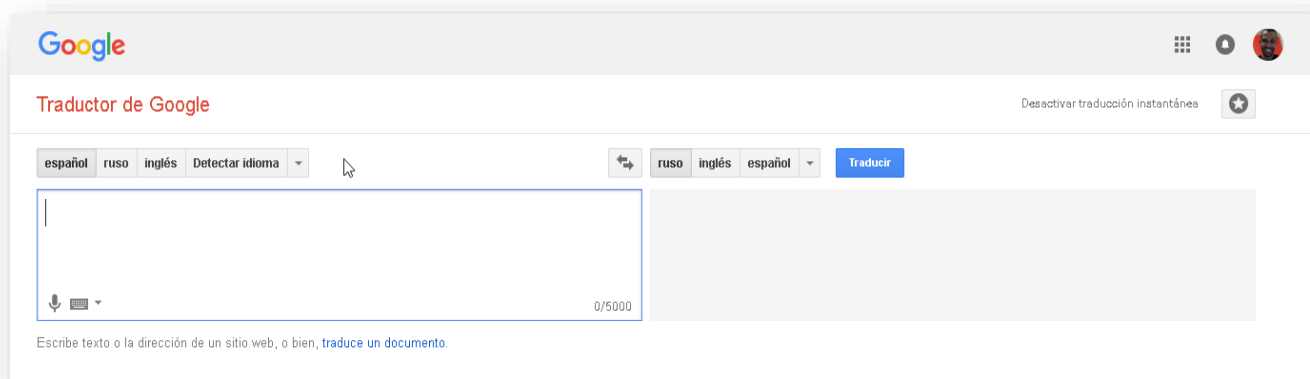
Ahora, identificaremos los objetos de nuestra página web. Cuando damos un breve vistazo, podemos observar que son los siguientes.



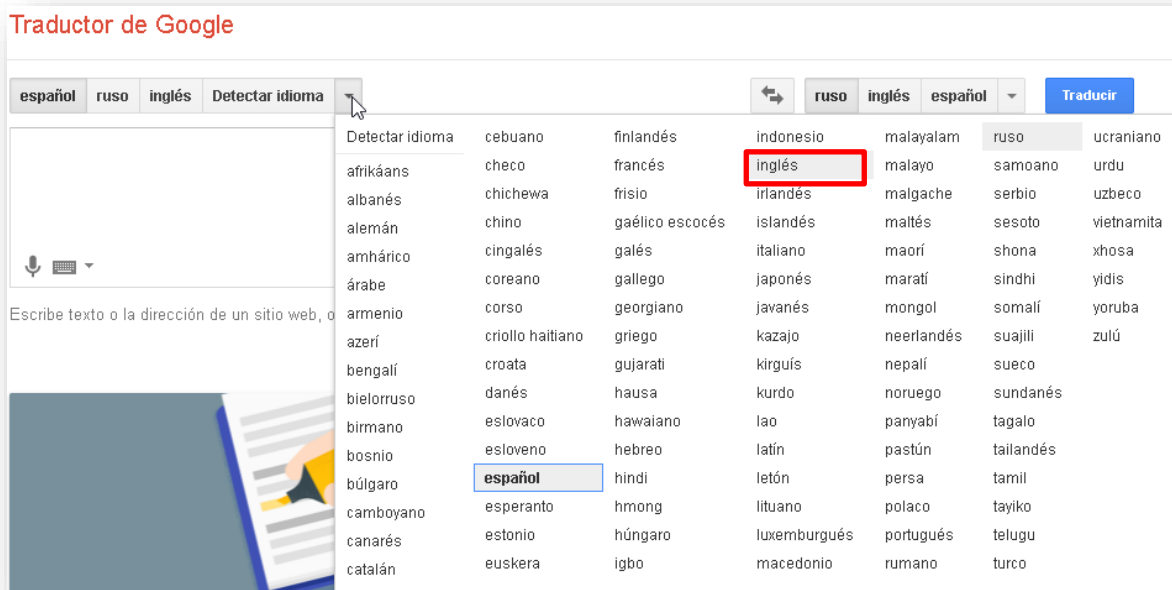
Se despliega un menú.



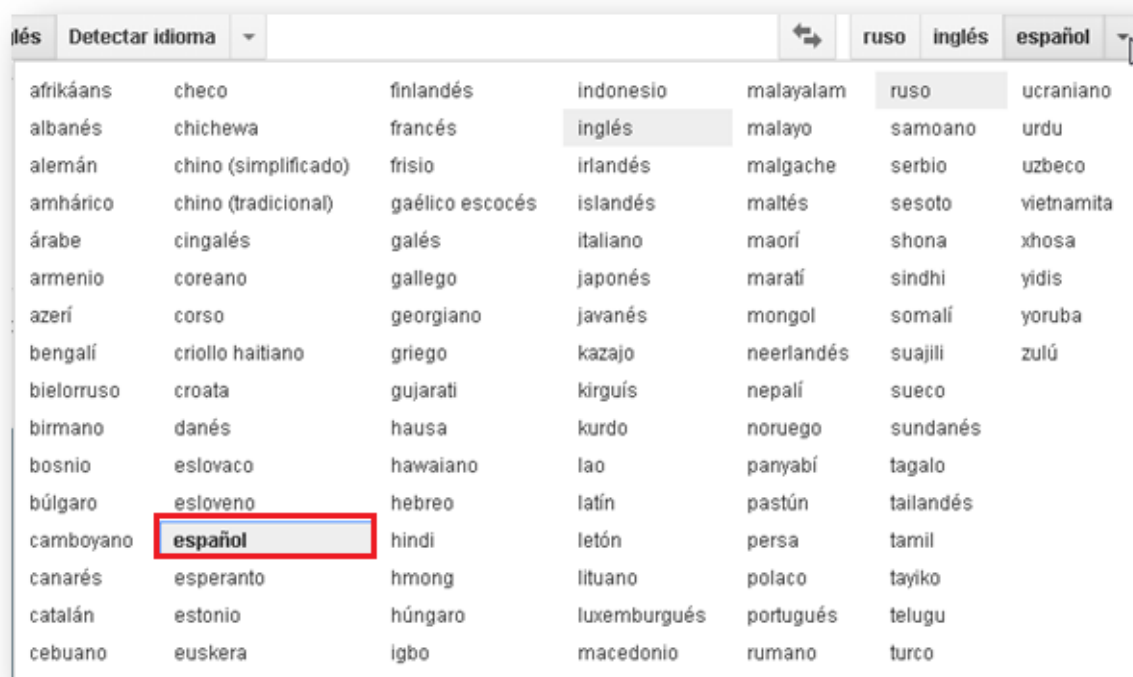
Nos lleva a una nueva ventana.



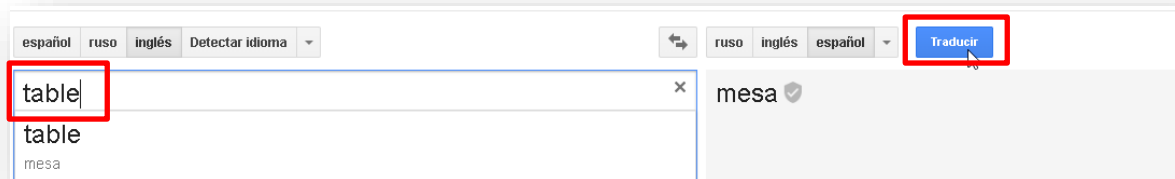
Escogemos el idioma Origen.



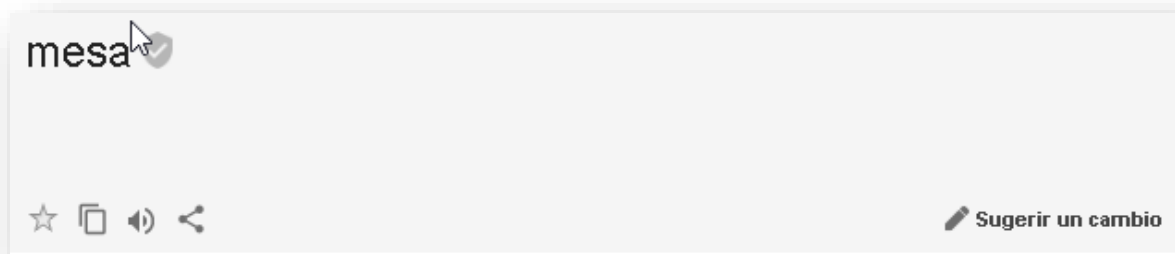
Escogemos el idioma destino.



Escribimos la palabra y le damos clic a traducir.



Verificamos que la palabra “mesa” aparece en el cuadro de texto resultante.



¡Identifiquemos todos los objetos!

En la página inicial de google, el primero que identificaremos será el botón que nos despliega el menú de aplicaciones, lo haremos del mismo modo que lo sabemos hacer hasta el día de hoy, clic derecho, 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 id = “gbwa”.

Iremos a nuestro paquete “co.com.proyectobase.screenplay.ui” y abriremos la clase “GoogleHomePage”. Y 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

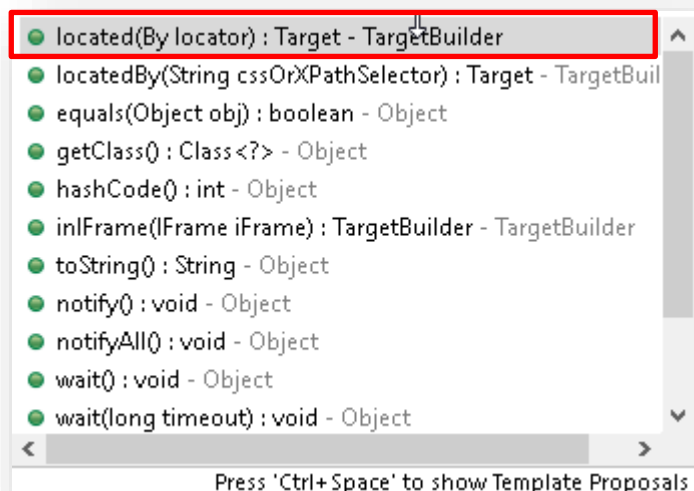
Para nuestra guía al botón de aplicaciones le llamaremos BOTON_APLICACIONES. Para instanciar un objeto de tipo Target completaremos la línea con el siguiente código.

```
public static final Target BOTON_APLICACIONES = Target.  
    the(String targetElementName) : TargetBuilder - Target
```

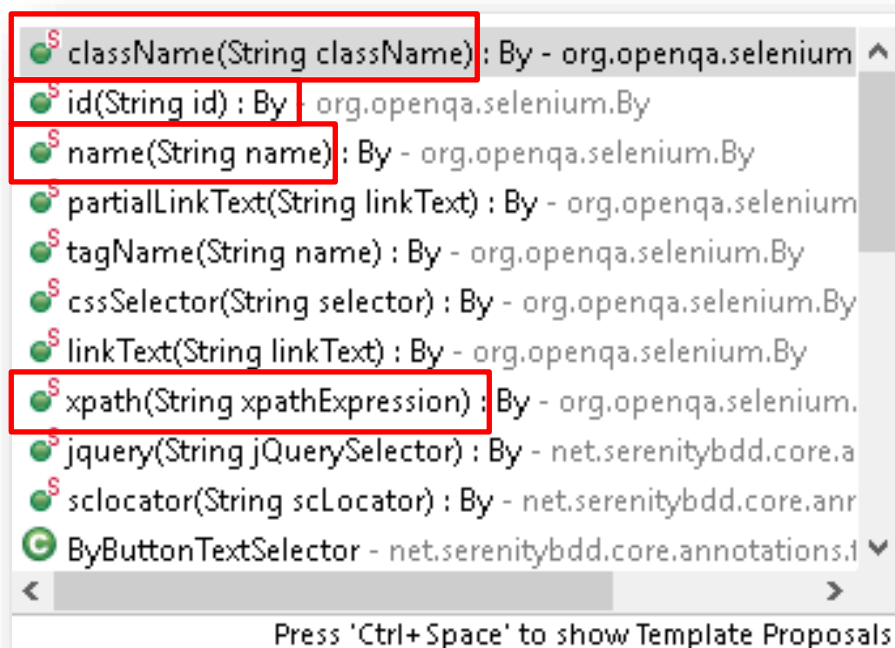
```
public static final Target BOTON_APLICACIONES = Target.the("El botón que  
muestra las aplicaciones").
```



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”.



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



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 identifique el objeto.



En nuestro caso inspeccionamos el objeto con un id = "gbwa". Nuestra línea de código queda de la siguiente forma:

```
public static final Target BOTON_APLICACIONES = Target.the("El botón que muestra las aplicaciones").located(By.id("gbwa"));
```

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 este Page identificaremos también el botón del Traductor, la línea de código sería:

```
public static final Target BOTON_GOOGLE_TRANSLATE = Target.the("Botón de app traductor").located(By.id("gb51"));
```

Nuestra clase "**GoogleHomePage**" quedaría así:

```
1 package co.com.proyectobase.screenplay.ui;
2
3 import net.serenitybdd.core.annotations.findby.By;
4 import net.serenitybdd.core.pages.PageObject;
5 import net.serenitybdd.screenplay.targets.Target;
6 import net.thucydides.core.annotations.DefaultUrl;
7
8
9 @DefaultUrl("http://www.google.com/")
10 public class GoogleHomePage extends PageObject {
11
12     public static final Target BOTON_APLICACIONES = Target.the("El boton que muestra las aplicaciones")
13         .located(By.id("gbwa"));
14     public static final Target BOTON_GOOGLE_TRANSLATE = Target.the("Botón de app traductor")
15         .located(By.id("gb51"));
16
17 }
```



Ahora vamos a Crear un nuevo page llamado “**GoogleTraductorPage**” en el paquete “**co.com.proyectobase.screenplay.ui**” e identifica los objetos descritos para traducir una palabra y asigne los siguientes nombres:

- BOTON LENGUAJE_ORIGEN.
- BOTON LENGUAJE_DESTINO.
- OPCION_ESPANOL.
- OPCION_INGLES.
- AREA_DE_TRADUCCION
- BOTON_TRADUCIR.
- AREA_TRADUCIDA.

¡Tú puedes!



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

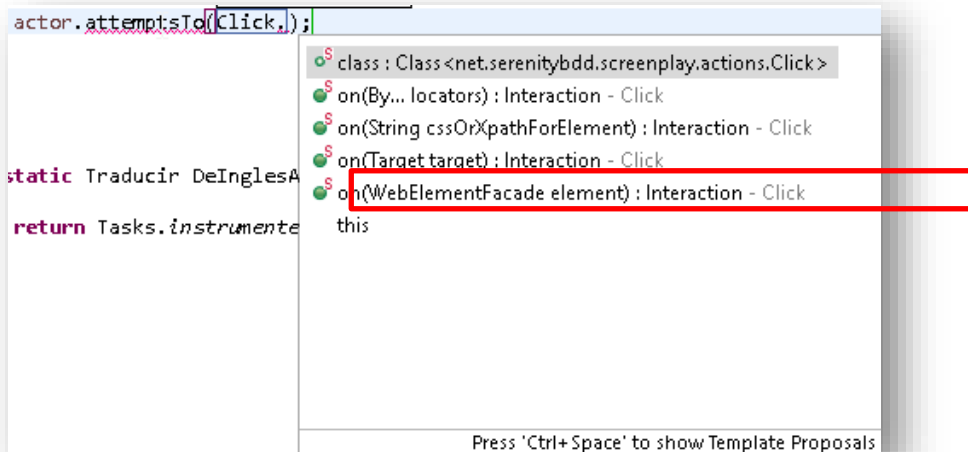
```
1 package co.com.proyectobase.screenplay.ui;
2
3 import net.serenitybdd.core.annotations.findby.By;
4 import net.serenitybdd.screenplay.targets.Target;
5
6 public class GoogleTraductorPage {
7
8     public static final Target BOTON_LINGUAJE_ORIGEN = Target.the("Botón del idioma origen")
9         .located(By.id("gt-sl-gms"));
10    public static final Target BOTON_LINGUAJE_DESTINO = Target.the("Botón del idioma destino")
11        .located(By.id("gt-tl-gms"));
12    public static final Target OPCION_INGLES = Target.the("La opción inglés")
13        .located(By.xpath("//div[@id='gt-sl-gms-menu']/table/tbody/tr/td/div[contains(text(),'ingl')]"));
14    public static final Target OPCION_ESPANOL = Target.the("El segundo idioma")
15        .located(By.xpath("//div[@id='gt-tl-gms-menu']/table/tbody/tr/td/div[contains(text(),'espa')]"));
16    public static final Target AREA_DE_TRADUCCION = Target.the("El lugar donde se escriben las palabras a traducir")
17        .located(By.id("source"));
18    public static final Target BOTON_TRADUCIR = Target.the("El botón traducir")
19        .located(By.id("gt-lang-submit"));
20    public static final Target AREA_TRADUCIDA = Target.the("El área donde ya se presenta la palabra traducida")
21        .located(By.id("gt-res-dir-ctr"));
22
23 }
24 }
```

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



```
1 package co.com.projectobase.screenplay.tasks;
2
3 import net.serenitybdd.screenplay.Actor;
4
5
6
7 public class Traducir implements Task{
8
9
10
11     @Override
12     public <T extends Actor> void performAs(T actor) {
13
14
15     }
16
17
18 public static Traducir DeInglesAEspanolLa(String palabra) {
19
20     return Tasks.instrumented(Traducir.class);
21 }
22
23 }
24 }
```

Recuerda que todo nuestro código irá siempre en el método performAs. Iniciaremos escribiendo actor.attemptsTo() y haremos cada paso desde que presionamos aplicaciones hasta que traducimos, una línea a la vez.



Escribiremos Click.on Y le pasaremos como parámetro el page que contiene al objeto deseado (GoogleHomePage) seguido de un punto, con el nombre del objeto que compete (BOTON_APLICACIONES).

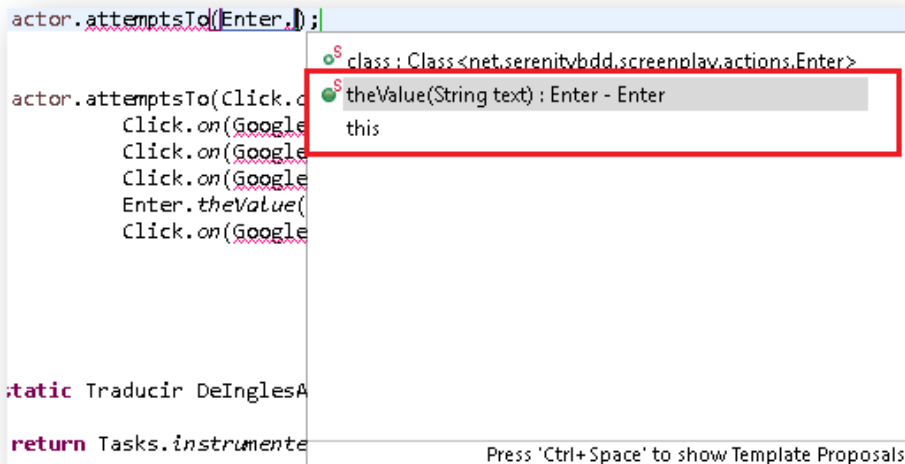


```
actor.attemptsTo(Click.on(GoogleHomePage.BOTON_APLICACIONES));  
actor.attemptsTo(Click.on(GoogleHomePage.BOTON_GOOGLE_TRANSLATE));
```

Añadimos todas las instrucciones que se requieren para lograr la traducción:

```
actor.attemptsTo(Click.on(GoogleTraductorPage.BOTON_LENGUAJE_ORIGEN));  
actor.attemptsTo(Click.on(GoogleTraductorPage.OPCION_INGLES));  
actor.attemptsTo(Click.on(GoogleTraductorPage.BOTON_LENGUAJE_DESTINO));  
actor.attemptsTo(Click.on(GoogleTraductorPage.OPCION_ESPANOL));
```

De esta manera interactuamos con objetos a los cuales se les puede realizar la acción de dar clic, para escribir en campos entonces usaremos el método Enter, y lo haremos de la siguiente manera:



```
actor.attemptsTo(Enter.);  
  
actor.attemptsTo(Click.on(GoogleTraductorPage.BOTON_LENGUAJE_ORIGEN));  
Click.on(GoogleTraductorPage.OPCION_INGLES);  
Click.on(GoogleTraductorPage.BOTON_LENGUAJE_DESTINO);  
Enter.theValue("Ingles");  
Click.on(GoogleTraductorPage.OPCION_ESPANOL);  
  
static Traducir DeInglesAEspañol()  
  
return Tasks.instrumented(Enter.class, "Enter", this);
```

Press 'Ctrl+Space' to show Template Proposals



```
actor.attemptsTo(Enter.theValue(palabra).into(GoogleTraductorPage.AREA_DE_TRADUCCION));
```

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

```
private String palabra;
```

Por último, agregamos el último clic en el botón traducir.

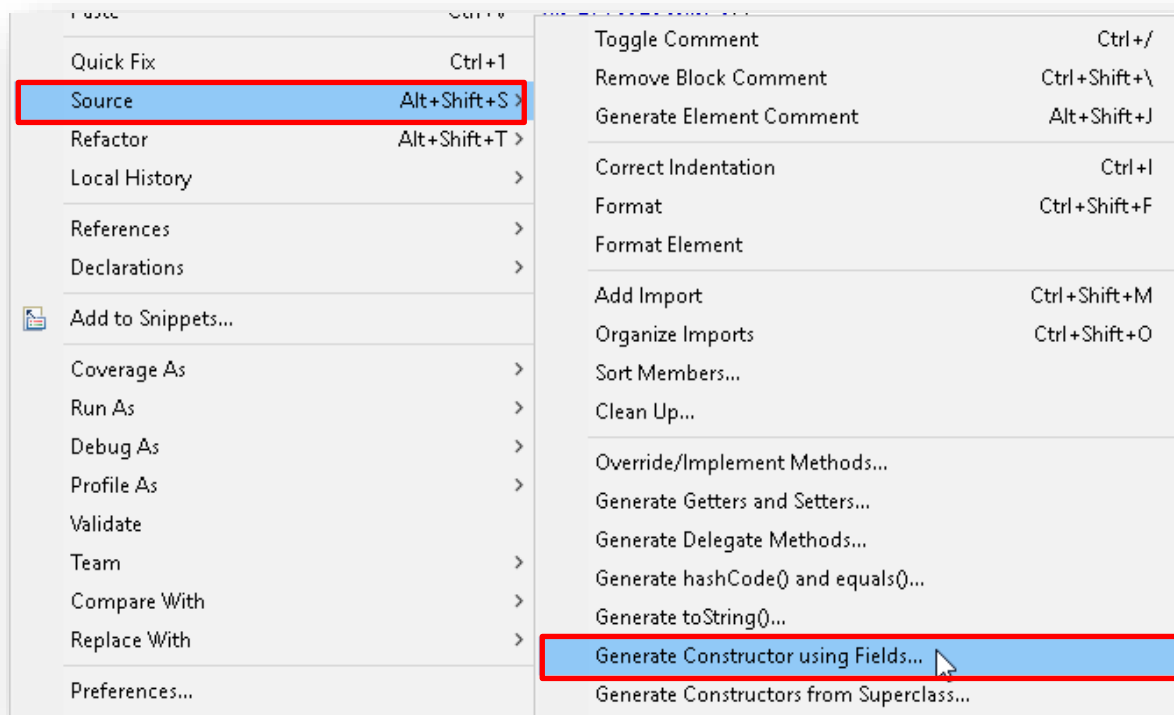
```
actor.attemptsTo(Click.on(GoogleTraductorPage.BOTON_TRADUCIR));
```

Nuestra clase Traducir, debe lucir de la siguiente manera:

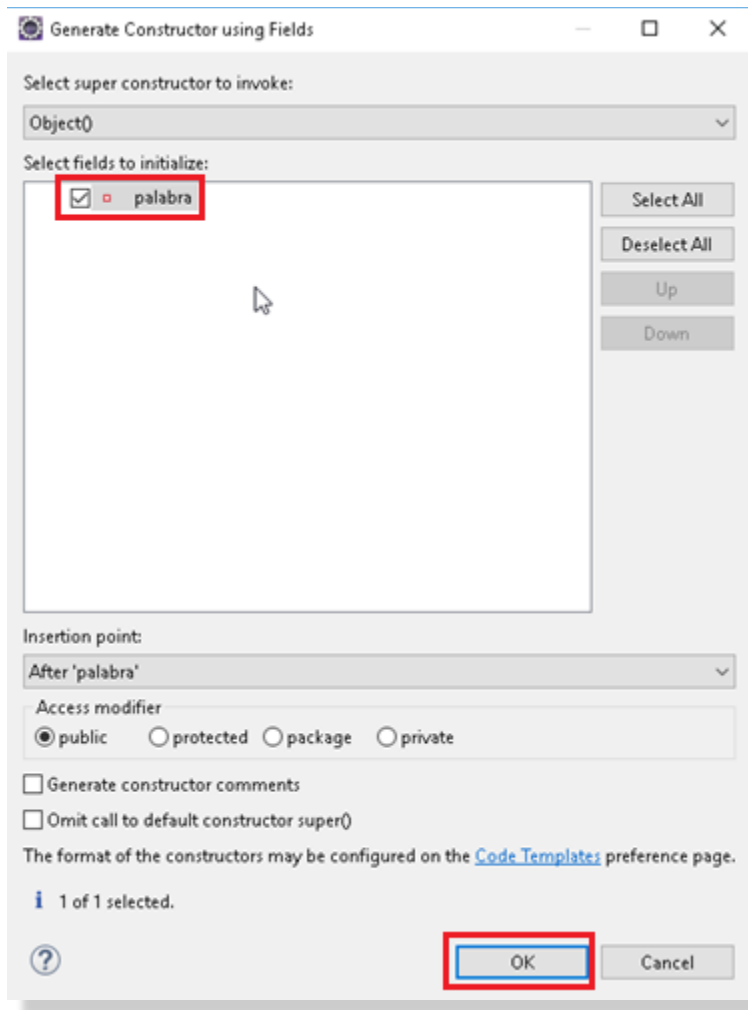
```
1 package co.com.proyectobase.screenplay.tasks;
2
3 import co.com.proyectobase.screenplay.ui.GoogleHomePage;
10
11 public class Traducir implements Task{
12
13     private String palabra;
14
15     @Override
16     public <T extends Actor> void performAs(T actor) {
17
18         actor.attemptsTo(Click.on(GoogleHomePage.BOTON_APLICACIONES));
19         actor.attemptsTo(Click.on(GoogleHomePage.BOTON_GOOGLE_TRANSLATE));
20
21         actor.attemptsTo(Click.on(GoogleTraductorPage.BOTON_LENGUAJE_ORIGEN));
22         actor.attemptsTo(Click.on(GoogleTraductorPage.OPCION_INGLES));
23         actor.attemptsTo(Click.on(GoogleTraductorPage.BOTON_LENGUAJE_DESTINO));
24         actor.attemptsTo(Click.on(GoogleTraductorPage.OPCION_ESPANOL));
25         actor.attemptsTo(Enter.theValue(palabra).into(GoogleTraductorPage.AREA_DE_TRADUCCION));
26
27         actor.attemptsTo(Click.on(GoogleTraductorPage.BOTON_TRADUCIR));
28
29     }
30
31
32
33 public static Traducir DeInglesAEspanolLa(String palabra) {
34
35     return Tasks.instrumented(Traducir.class);
36 }
37
38 }
39
```

Debido a que esta clase cuenta, con un atributo de tipo String llamado “palabra” 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 Constructor using Fields...”.



Seleccionamos el atributo que nos aparece en la ventana de creación. Y presionamos OK.



Por último, verificamos que el constructor se haya creado correctamente.

```
public Traducir(String palabra) {  
    super();  
    this.palabra = palabra;  
}
```



La clase completa, luciría de la siguiente forma:

```
1 package co.com.proyectobase.screenplay.tasks;
2
3 import co.com.proyectobase.screenplay.ui.GoogleHomePage;
10
11 public class Traducir implements Task{
12
13     private String palabra;
14
15     public Traducir(String palabra) {
16         super();
17         this.palabra = palabra;
18     }
19
20     @Override
21     public <T extends Actor> void performAs(T actor) {
22
23         actor.attemptsTo(Click.on(GoogleHomePage.BOTON_APLICACIONES));
24         actor.attemptsTo(Click.on(GoogleHomePage.BOTON_GOOGLE_TRANSLATE));
25         actor.attemptsTo(Click.on(GoogleTraductorPage.BOTON_LENGUAJE_ORIGEN));
26         actor.attemptsTo(Click.on(GoogleTraductorPage.OPCION_INGLES));
27         actor.attemptsTo(Click.on(GoogleTraductorPage.BOTON_LENGUAJE_DESTINO));
28         actor.attemptsTo(Click.on(GoogleTraductorPage.OPCION_ESPANOL));
29         actor.attemptsTo(Enter.theValue(palabra).into(GoogleTraductorPage.AREA_DE_TRADUCCION));
30         actor.attemptsTo(Click.on(GoogleTraductorPage.BOTON_TRADUCIR));
31
32     }
33
34     public static Traducir DeInglesAEspanolLa(String palabra) {
35         return Tasks.instrumented(Traducir.class);
36     }
37 }
38
```

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

```
11 public class Traducir implements Task{
12
13     private String palabra;
14
15     public Traducir(String palabra) {
16         super();
17         this.palabra = palabra;
18     }
19
20     @Override
21     public <T extends Actor> void performAs(T actor) {
22
23         actor.attemptsTo(Click.on(GoogleHomePage.BOTON_APLICACIONES));
24         actor.attemptsTo(Click.on(GoogleHomePage.BOTON_GOOGLE_TRANSLATE));
25         actor.attemptsTo(Click.on(GoogleTraductorPage.BOTON_LENGUAJE_ORIGEN));
26         actor.attemptsTo(Click.on(GoogleTraductorPage.OPCION_INGLES));
27         actor.attemptsTo(Click.on(GoogleTraductorPage.BOTON_LENGUAJE_DESTINO));
28         actor.attemptsTo(Click.on(GoogleTraductorPage.OPCION_ESPANOL));
29         actor.attemptsTo(Enter.theValue(palabra).into(GoogleTraductorPage.AREA_DE_TRADUCCION));
30         actor.attemptsTo(Click.on(GoogleTraductorPage.BOTON_TRADUCIR));
31
32     }
33
34     public static Traducir DeInglesAEspanolLa(String palabra) {
35         return Tasks.instrumented(Traducir.class, palabra);
36     }
37 }
38
```



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.**

```
TEST PASSED: Traducir de Ingles a Español
```

```
[main] INFO net.serenitybdd.core.Serenity -
```

```
  3-] TEST PASSED
```

```
TEST PASSED: Traducir de Ingles a Español
```

¡NOS VEMOS EN LA PRÓXIMA GUÍA!

