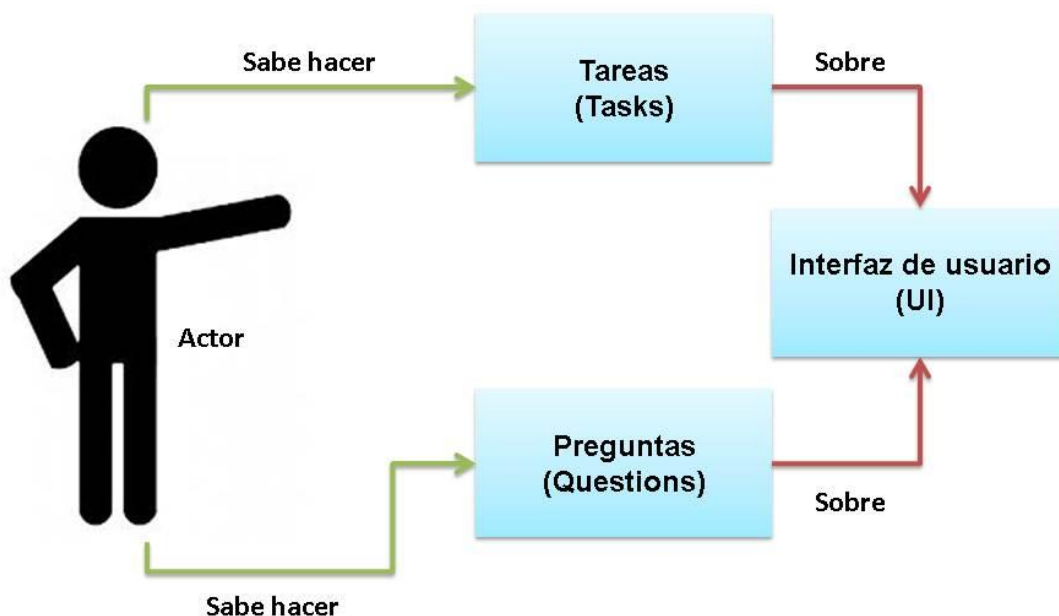


3

SERENITY BDD + SCREENPLAY con CUCUMBER

Primeros pasos con el patrón

Screenplay

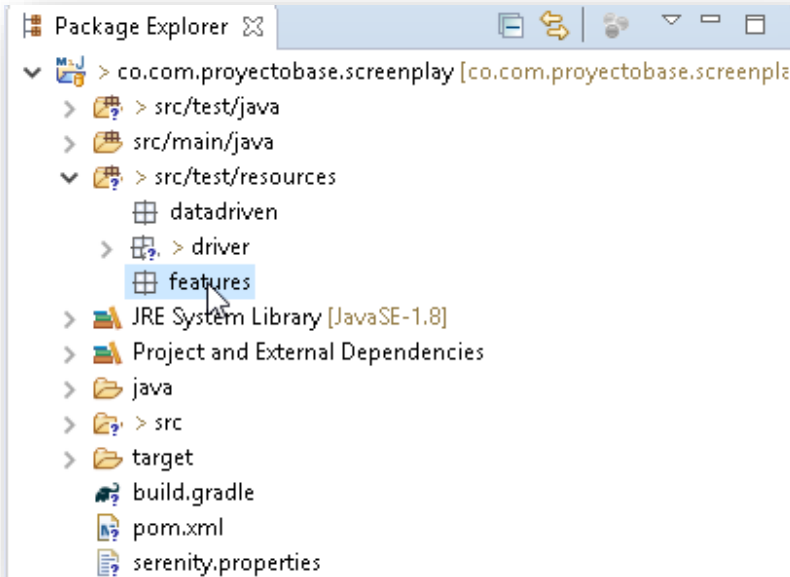


En la presente guía, daremos nuestros primeros pasos sobre el patrón Screenplay. Una vez cargado nuestro proyecto base en eclipse, procederemos a crear una historia de usuario.

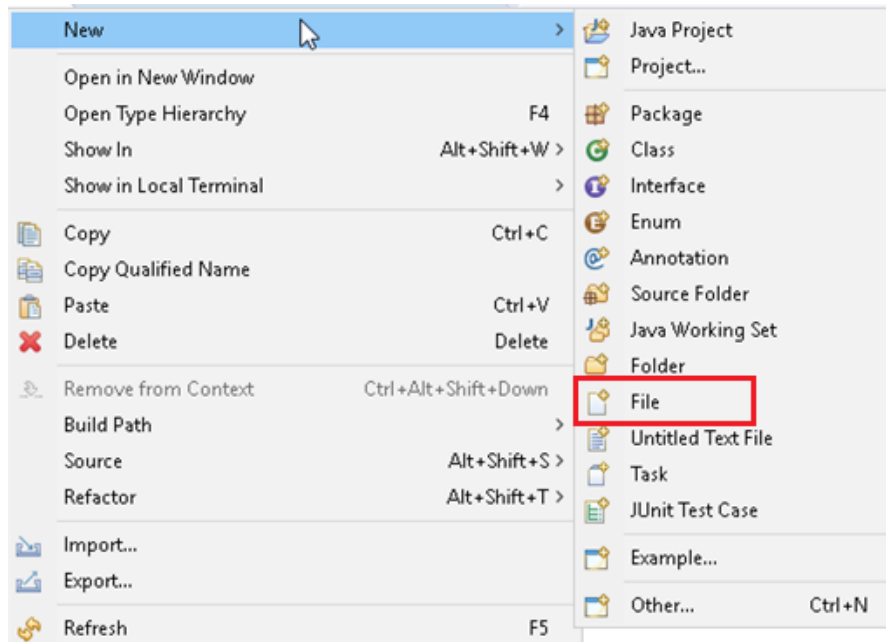


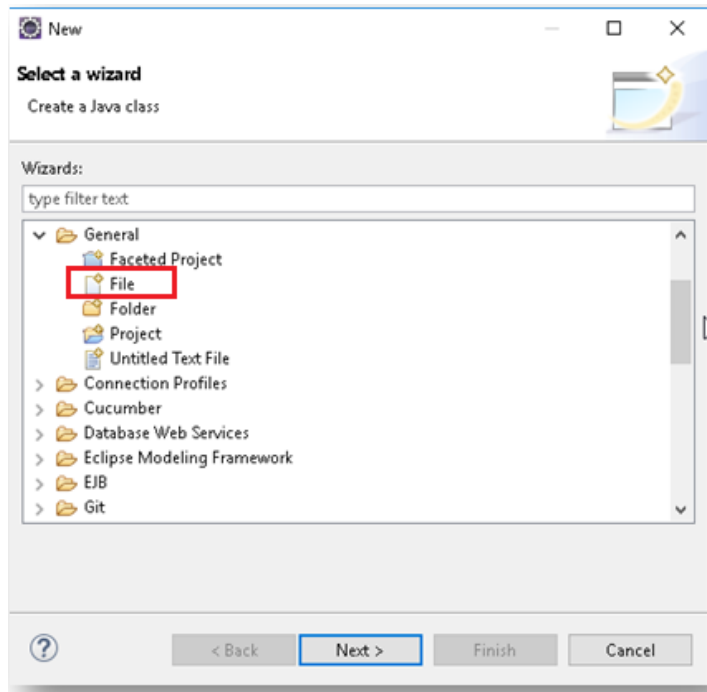
1. Crear una historia de usuario “.feature”

Para esto, iremos a nuestro proyecto y en la carpeta “src/test/resources” encontraremos un paquete llamado “features”.



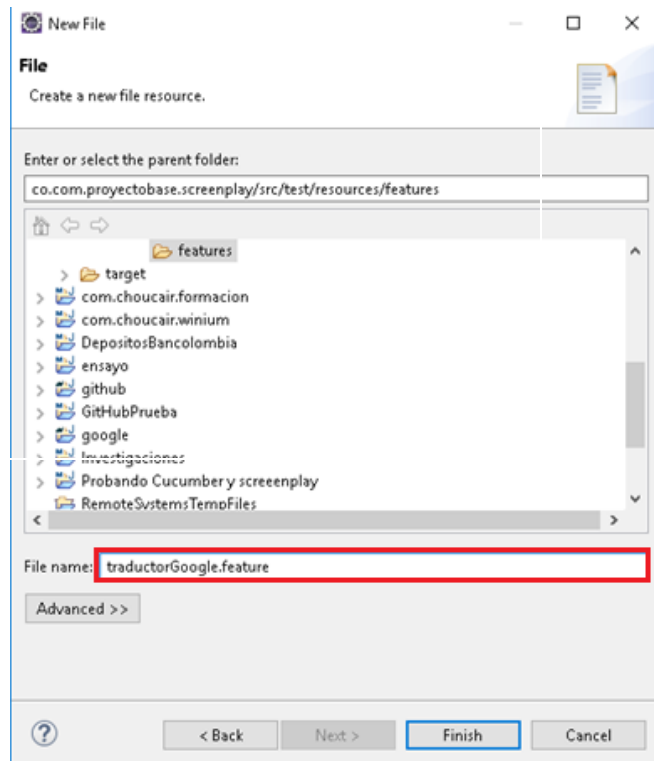
Una vez ahí, presionamos clic derecho sobre el paquete “features”, vamos a New y luego en File.



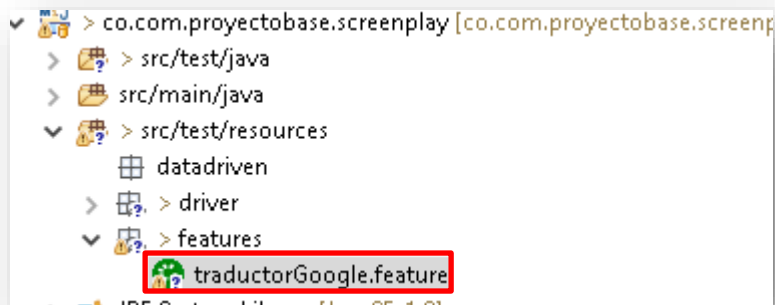


En caso de que **NO** aparezca la opción File en la ventana desplegada por New, entonces ir a “Other...”, y en la nueva ventana que aparece ir a la carpeta “General” y escoger la opción “File”.

Al escoger la opción File, se desplegará una nueva ventana para crear un archivo “.feature”, para esta guía, crearemos uno llamado “traductorGoogle.feature”. **Recuerda** que debes añadirle la extensión “.feature” al final.



Por último, damos clic en el botón “Finish” y verificamos en nuestro proyecto la creación de nuestra feature.



```
TraductorGoogle.feature
1  #Author: your.email@your.domain.com
2  #Keywords Summary :
3  #Feature: List of scenarios.
4  #Scenario: Business rule through list of steps with arguments.
5  #Given: Some precondition step
6  #When: Some key actions
7  #Then: To observe outcomes or validation
8  #And,But: To enumerate more Given,When,Then steps
9  #Scenario Outline: List of steps for data-driven as an Examples and <placeholder>
10 #Examples: Container for s table
11 #Background: List of steps run before each of the scenarios
12 #""" (Doc Strings)
13 #| (Data Tables)
14 #@ (Tags/Labels):To group Scenarios
15 #<> (placeholder)
16 #""
17 ## (Comments)
18 #Sample Feature Definition Template
19 @tag
20 Feature: Title of your feature
21   I want to use this template for my feature file
22
23   @tag1
24   Scenario: Title of your scenario
25     Given I want to write a step with precondition
26     And some other precondition
27     When I complete action
28     And some other action
29     And yet another action
30     Then I validate the outcomes
31     And check more outcomes
32
```

Si tenemos el editor de Cucumber correctamente instalado, al abrir nuestra feature lucirá como la imagen anterior. Para esta guía, crearemos una historia de usuario que describa a una persona que abre un navegador, entra a las aplicaciones de google, escoge el traductor, escoge dos idiomas, el primero inglés, el segundo español, escribe una palabra en Inglés, da clic en el botón “traducir” y verifica la palabra traducida al español.



2. Escribir la historia de usuario

Para iniciar, es recomendable borrar las siguientes líneas de la feature

```
1 #Author: your.email@your.domain.com
2 #Keywords Summary :
3 #Feature: List of scenarios.
4 #Scenario: Business rule through list of steps with arguments.
5 #Given: Some precondition step
6 #When: Some key actions
7 #Then: To observe outcomes or validation
8 #And,But: To enumerate more Given,When,Then steps
9 #Scenario Outline: List of steps for data-driven as an Examples and <placeholder>
10 #Examples: Container for s table
11 #Background: List of steps run before each of the scenarios
12 #""" (Doc Strings)
13 #| (Data Tables)
14 #@ (Tags/Labels):To group Scenarios
15 #<> (placeholder)
16 #"""
17 ## (Comments)
18 #Sample Feature Definition Template
```

Dejando solo la línea que hace alusión al correo del autor, ahí podemos escribir nuestro correo, que evidencie nuestra autoría. Podemos también eliminar el resto del texto y escribir a continuación nuestra historia de usuario, así como se describe a continuación. La idea es que nuestra Feature se escriba en un único idioma, es decir, toda en Ingles o en español, no debemos mezclar distintitos idiomas en la misma.

Para que sea reconocido un idioma diferente al Ingles deberíamos colocar en los Tag iniciales lo siguiente:

```
2 # language:es
```

Con esto estamos determinando que el idioma de nuestra feature es Español.

Gherkin Dialects										
Language	Feature	Background	Scenario	Scenario Outline	Examples	Given	When	Then	And	But
Spanish (es)	Característica	Antecedentes	Escenario	Esquema del escenario	Ejemplos	*	*	*	*	*
						Dado	Cuando	Entonces	Y	Pero
						Dada			E	
						Dados				
						Dadas				

Feature en Español

```
1 #Author: varias@choucairtesting.com
2 # language:es
3 Característica: Traductor de Google
4   Como usuario
5   Quiero ingresar al traductor de Google
6   A Traducir palabras entre diferentes lenguajes.
7
8 Escenario: Traducir de Inglés a Español
9   Dado que Rafa quiere usar el Traductor de Google
10  Cuando el traduce la palabra table de Inglés a Español
11  Entonces el debería ver la palabra mesa en la pantalla
```

Feature en Inglés

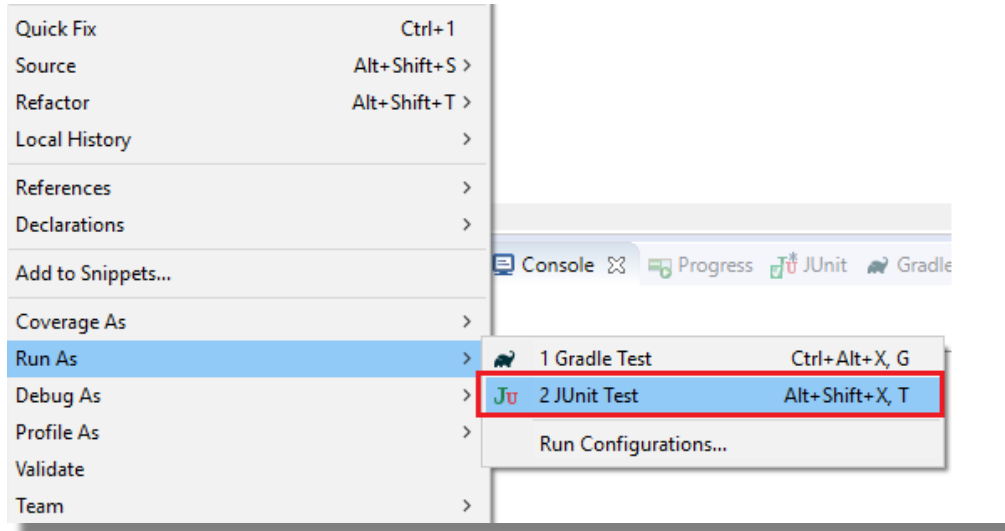
```
1 #Author: varias@choucairtesting.com
2 Feature: Google Translate
3   As a web user
4   I want to use Google Translate
5   to translate words between different languages
6
7 Scenario: Translate from Source Language to Target Language
8   Given that Rafa want to use Google Translate
9   When he translate the word table from inglés to español
10  Then he should see the word Mesa on the screen
```

Notamos, que a diferencia del patrón “Page Object Model”, ahora nuestras acciones las ejecutará un **actor**, en este ejemplo “Rafa” es nuestro actor, y es el que ejecuta todas las acciones de esta prueba. En tu ejemplo, puedes ponerle el nombre que estimes conveniente al actor.



3. Generar los métodos que deben ser agregados a la StepDefinition

Para generar los métodos que se requieren para la StepDefinition, parametrizamos nuestra clase "RunnerTags" le daremos clic derecho sobre ésta, y en la opción "Run as" escogemos "JUnit Test".



Veremos que en la consola nos generará unos métodos sugeridos para copiar y pegar en nuestra clase StepDefinition

```
1 Scenarios (@[33m1 undefined@[0m)
3 Steps (@[33m3 undefined@[0m)
0m1,297s

You can implement missing steps with the snippets below:

@Dado("^que Rafa quiere usar el Traductor de Google$")
public void queRafaQuiereUsarElTraductorDeGoogle() throws Exception {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

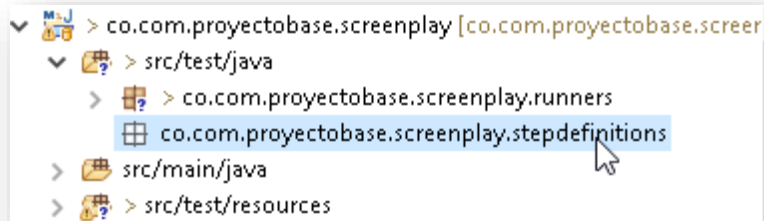
@Cuando("^el traduce la palabra table de Inglés a Español$")
public void elTraduceLaPalabraTableDeInglésAEspañol() throws Exception {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@Entonces("^el deberia ver la palabra mesa en la pantalla$")
public void elDeberiaVerLaPalabraMesaEnLaPantalla() throws Exception {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}
```

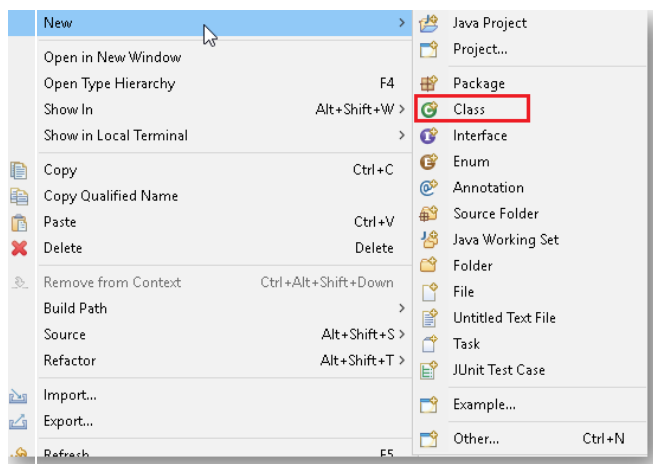


4. Crear clase StepDefinition

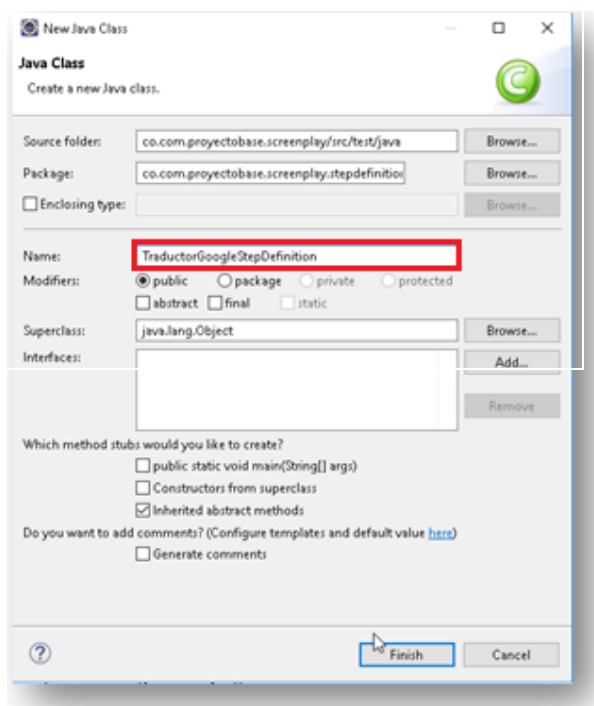
Una vez se hayan generado los métodos sugeridos, los copiaremos, e iremos a nuestra carpeta “src/test/java” y en el paquete “co.com.projectobase.screenplay.stepdefinitions” crearemos una clase llamada “TraductorGoogleStepDefintions”.



New>Class



Escribimos el nombre de la clase “TraductorGoogleStepDefintions” y clic en el botón Finish



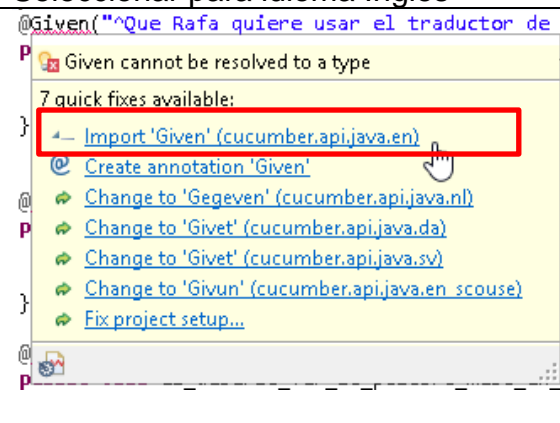
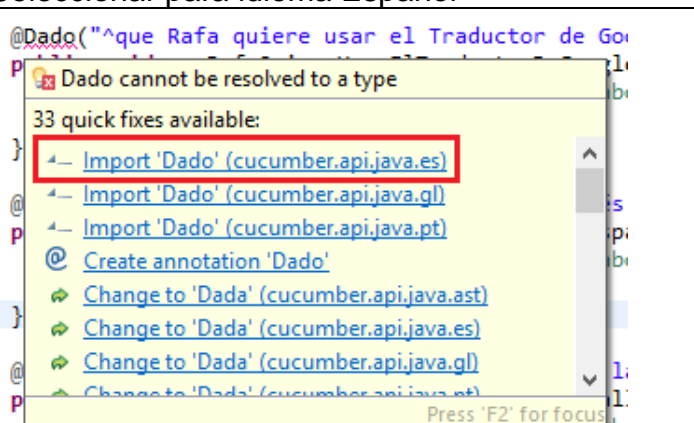
Finalmente, pegamos en esta clase los métodos sugeridos anteriormente.

```
@Given("^Que Rafa quiere usar el traductor de google$")
public void queRafaQuiereUsarElTraductorDeGoogle() throws Exception {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@When("^el traduce la palabra table del inglés al español$")
public void elTraduceLaPalabraTableDelInglésAlEspañol() throws Exception {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@Then("^el debería ver la palabra mesa en la pantalla$")
public void elDeberiaVerLaPalabraMesaEnLaPantalla() throws Exception {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}
```

Importamos las librerías para las annotations (@Given, @When y @Then) o (@Dado, @Cuando y @Entonces).

Seleccionar para idioma Ingles	Seleccionar para idioma Español
	

Y por último eliminamos el código sugerido dentro de los métodos. Como resultado, tendremos una clase, que lucirá como la siguiente imagen.



```
TraductorGoogleStepDefinitions.java
1 package co.com.proyectobase.screenplay.stepdefinitions;
2
3 import cucumber.api.java.en.Given;
4 import cucumber.api.java.en.Then;
5 import cucumber.api.java.en.When;
6
7 public class TraductorGoogleStepDefinitions {
8
9     @Given("^Que Rafa quiere usar el traductor de google$")
10     public void queRafaQuiereUsarElTraductorDeGoogle() throws Exception {
11
12     }
13
14     @When("^el traduce la palabra table del inglés al español$")
15     public void elTraduceLaPalabraTableDelInglésAlEspañol() throws Exception {
16
17     }
18
19     @Then("^el deberia ver la palabra mesa en la pantalla$")
20     public void elDeberiaVerLaPalabraMesaEnLaPantalla() throws Exception {
21
22     }
23
24 }
```

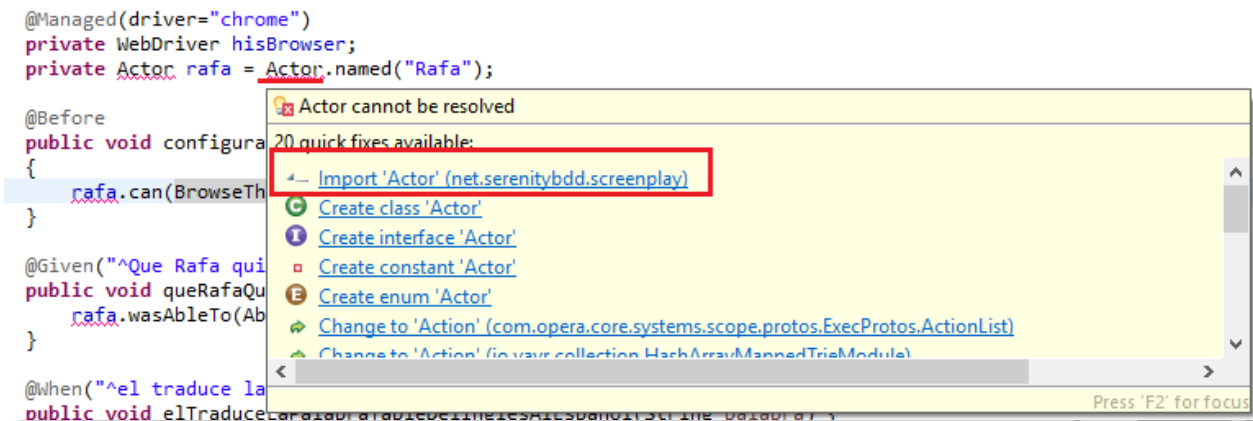
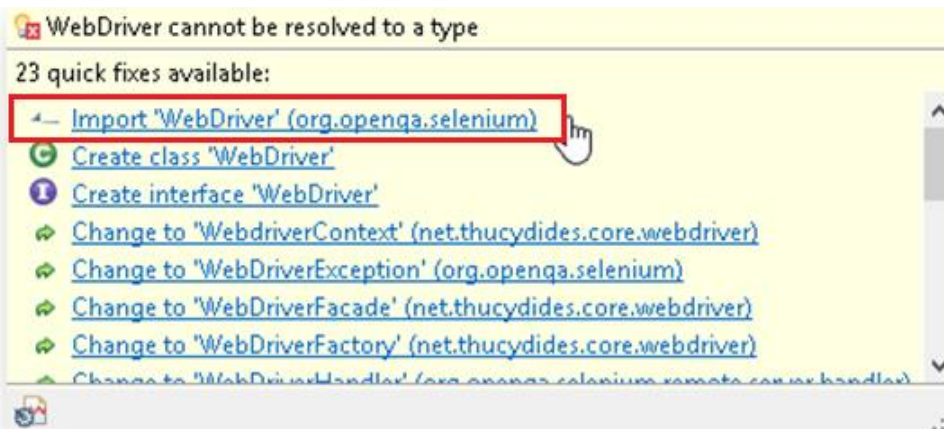
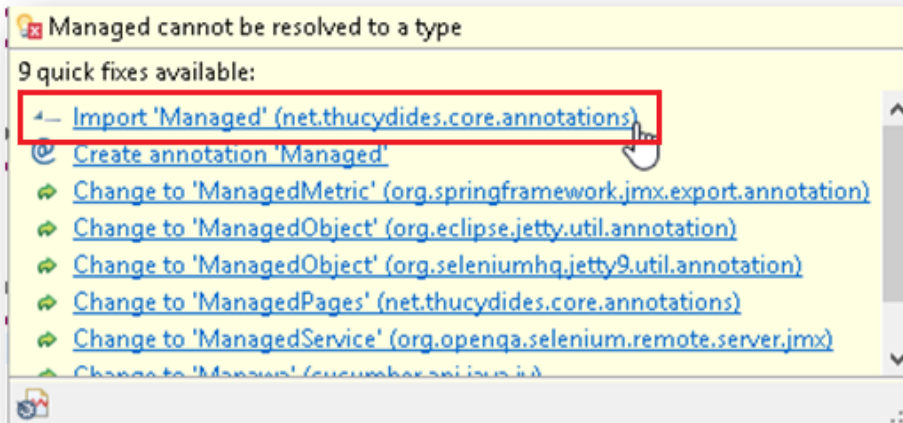
5. Crear Actores y configuración Inicial

Al inicio de nuestra clase TraductorGoogleStepDefinition agregaremos las siguientes tres líneas de código:

```
TraductorGoogleStepDefinitions.java
1 package co.com.proyectobase.screenplay.stepdefinitions;
2
3 import cucumber.api.java.en.Given;
4 import cucumber.api.java.en.Then;
5 import cucumber.api.java.en.When;
6
7 public class TraductorGoogleStepDefinitions {
8
9     @Managed(driver="chrome")
10     private WebDriver hisBrowser;
11     private Actor rafa = Actor.named("Rafa");
12 }
```



Importamos las librerías para la annotation @Managed, Actor y para WebDriver.



Una vez importadas las librerías, nuestra clase, debe estar de la siguiente forma:

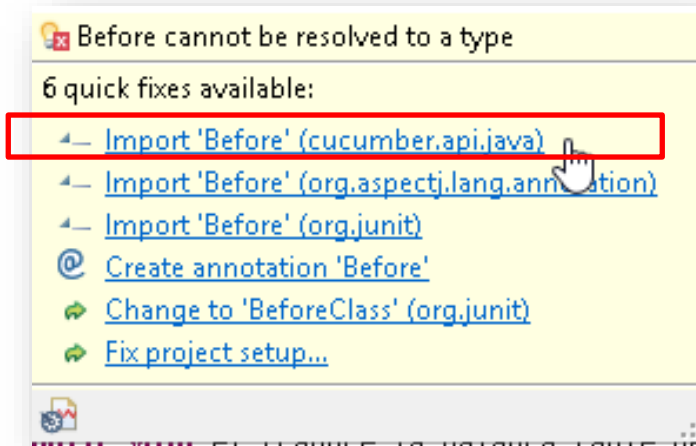
```
TraductorGoogleStepDefintions.java
1
2
3
4
5 import cucumber.api.java.en.Given;
6 import cucumber.api.java.en.Then;
7 import cucumber.api.java.en.When;
8 import net.serenitybdd.screenplay.Actor;
9 import net.thucydides.core.annotations.Managed;
10
11 public class TraductorGoogleStepDefintions {
12
13     @Managed(driver="chrome")
14     private WebDriver hisBrowser;
15     private Actor rafa = Actor.named("Rafa");
16
17
18     @Given("^Que Rafa quiere usar el traductor de google$")
19     public void queRafaQuiereUsarElTraductorDeGoogle() throws Exception {
20
21     }
22
23     @When("^el traduce la palabra table del inglés al español$")
24     public void elTraduceLaPalabraTableDelInglésAlEspañol() throws Exception {
25
26     }
27
28     @Then("^el debería ver la palabra mesa en la pantalla$")
29     public void elDeberiaVerLaPalabraMesaEnLaPantalla() throws Exception {
30
31     }
32
33 }
```

A continuación, crearemos un método que preparará la configuración inicial de nuestra prueba de automatización. Agreguemos las siguientes líneas de código en nuestra clase TraductorGoogleStepDefinition:

```
*TraductorGoogleStepDefintions.java
1
2
3
4
5 import cucumber.api.java.en.Given;
6 import cucumber.api.java.en.Then;
7 import cucumber.api.java.en.When;
8 import net.serenitybdd.screenplay.Actor;
9 import net.thucydides.core.annotations.Managed;
10
11 public class TraductorGoogleStepDefintions {
12
13     @Managed(driver="chrome")
14     private WebDriver hisBrowser;
15     private Actor rafa = Actor.named("Rafa");
16
17     @Before
18     public void configuracionInicial()
19     {
20
21     }
22 }
```



Agregamos las librerías que competen a la annotation @Before



Los actores que creamos tienen habilidades, y en nuestra clase TraductorGoogleStepDefintions, el actor es el gran protagonista. Entonces, en nuestro método de “configuracionInicial” le daremos la habilidad de navegar a nuestro actor con la siguiente línea de código:

```
*TraductorGoogleStepDefintions.java
import cucumber.api.java.en.Given;
6 import cucumber.api.java.en.Then;
7 import cucumber.api.java.en.When;
8 import net.serenitybdd.screenplay.Actor;
9 import net.thucydides.core.annotations.Managed;
10
11
12 public class TraductorGoogleStepDefintions {
13
14     @Managed(driver="chrome")
15     private WebDriver hisBrowser;
16     private Actor rafa = Actor.named("Rafa");
17
18     @Before
19     public void configuracionInicial()
20     {
21         rafa.can(BrowseTheWeb.with(hisBrowser));
22     }
23 }
```



La anterior línea, le da a nuestro actor la capacidad de navegar en la Web. Una de las características más interesantes que propone el patrón Screenplay es la manera en que se va creando código muy limpio y legible. Si leemos la línea anterior con algo de conocimiento de inglés, podemos leer “Rafa puede navegar en la web con su navegador” (Rafa can browse the web with his browser). Una vez agregada la línea de código propuesta, tendremos una clase parecida a la mostrada en la siguiente imagen:

```
TraductorGoogleStepDefintions.java  ✖
1  package co.com.projectobase.screenplay.stepdefinitions;
2
3+ import org.openqa.selenium.WebDriver;
12
13 public class TraductorGoogleStepDefintions {
14
15     @Managed(driver="chrome")
16     private WebDriver hisBrowser;
17     private Actor rafa = Actor.named("Rafa");
18
19     @Before
20     public void configuracionInicial()
21     {
22         rafa.can(BrowseTheWeb.with(hisBrowser));
23     }
24
25     @Given("^Que Rafa quiere usar el traductor de google$")
26     public void queRafaQuiereUsarElTraductorDeGoogle() throws Exception {
27
28     }
29
30     @When("^el traduce la palabra table del inglés al español$")
31     public void elTraduceLaPalabraTableDelInglésAlEspañol() throws Exception {
32
33     }
34
35     @Then("^el deberia ver la palabra mesa en la pantalla$")
36     public void elDeberiaVerLaPalabraMesaEnLaPantalla() throws Exception {
37
38     }
39
40 }
```

¡Felicidades has dado tus primeros pasos en el patrón Screenplay!

En nuestra próxima guía,
¡Crearemos nuestra primera tarea!

