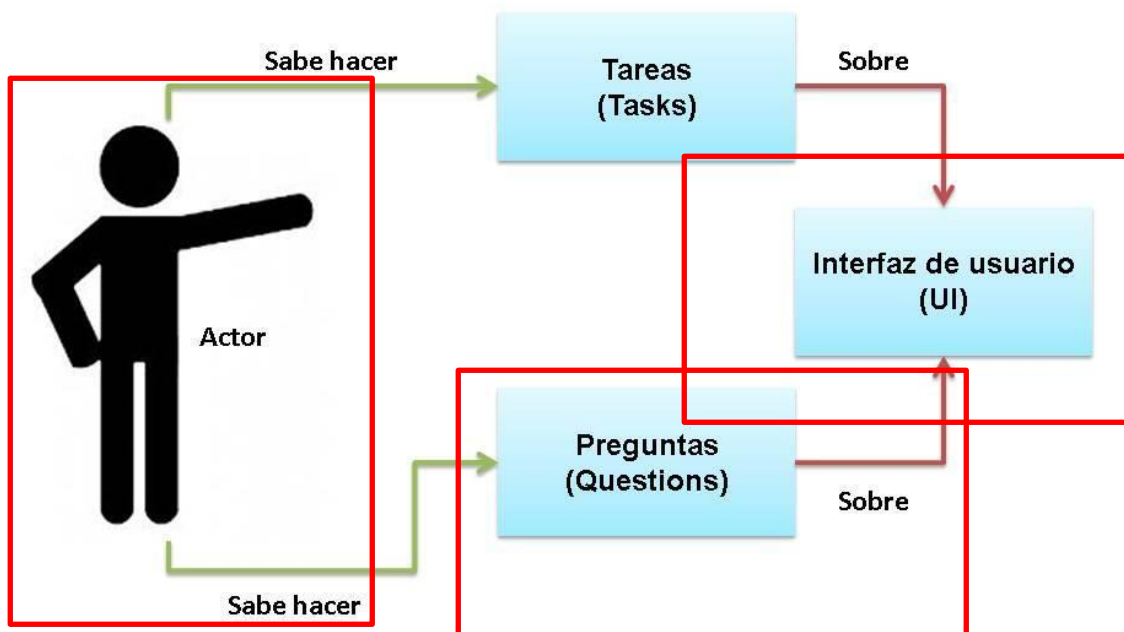


6 SERENITY BDD + SCREENPLAY con CUCUMBER

¡Bienvenidos a nuestra guía 6 del patrón Screenplay!

En esta guía Aprenderemos cómo implementar las verificaciones en el desarrollo de Questions.



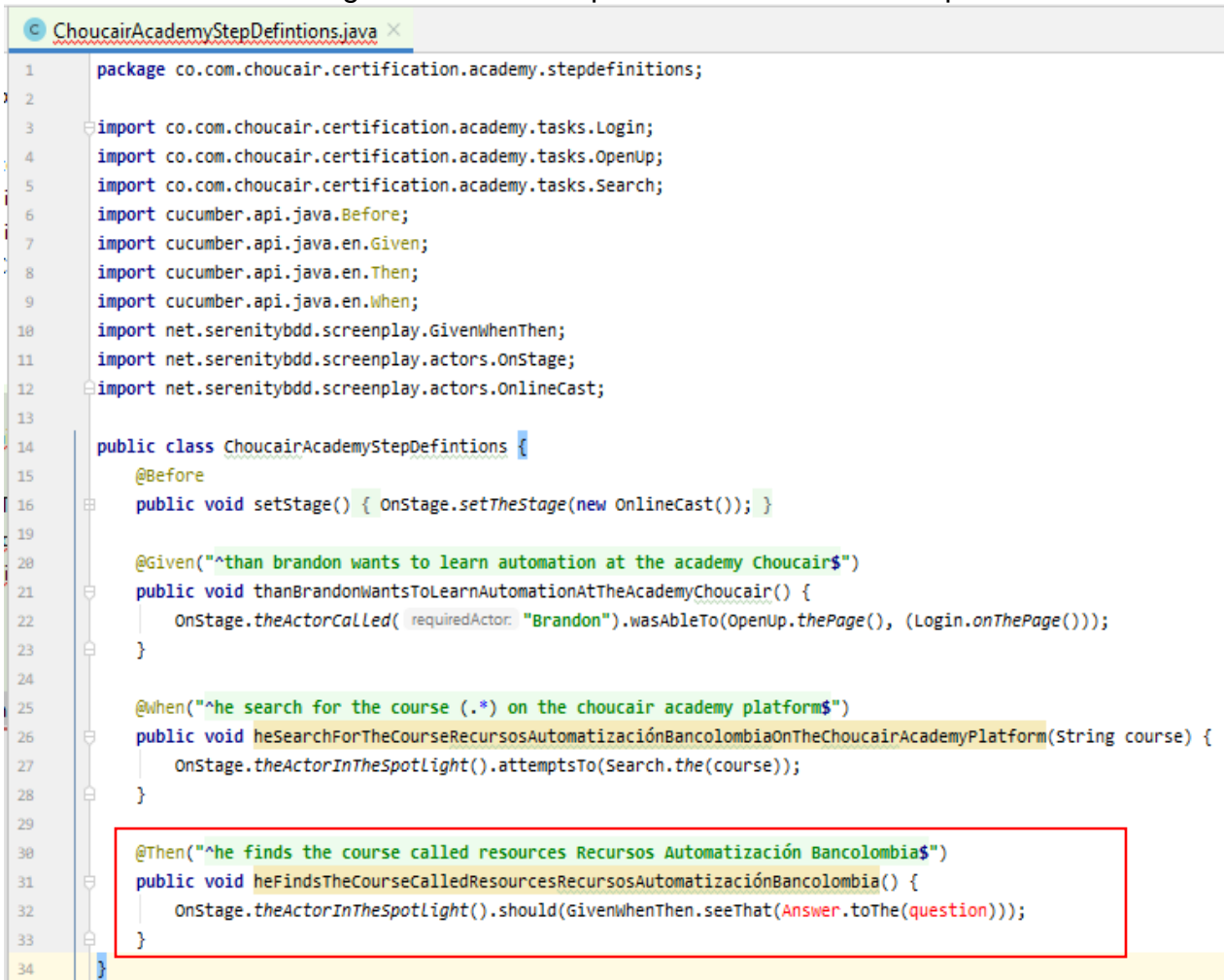
Una vez hayamos realizado nuestras tareas anteriores, terminaremos realizando la verificación de que toda nuestra historia de usuario salió conforme a lo esperado. En esta guía buscaremos darle implementación a la siguiente instrucción en nuestra historia de usuario.

Then he finds the course called resources Recursos Automatización Bancolombia

Entonces iremos nuevamente a nuestra clase “**ChoucairAcademyStepDefintions**”, en donde implementaremos el @Then.

```
@Then("^he finds the course called resources Recursos Automatización Bancolombia$")
public void heFindsTheCourseCalledResourcesRecursosAutomatizaciónBancolombia() {
    OnStage.theActorInTheSpotLight().should(GivenWhenThen.seeThat(Answer.toThe(question)));
}
```

la cual debe lucir de la siguiente forma después de las dos tareas implementadas.



```
ChoucairAcademyStepDefintions.java x
1 package co.com.choucair.certification.academy.stepdefinitions;
2
3 import co.com.choucair.certification.academy.tasks.Login;
4 import co.com.choucair.certification.academy.tasks.OpenUp;
5 import co.com.choucair.certification.academy.tasks.Search;
6 import cucumber.api.java.Before;
7 import cucumber.api.java.en.Given;
8 import cucumber.api.java.en.Then;
9 import cucumber.api.java.en.When;
10 import net.serenitybdd.screenplay.GivenWhenThen;
11 import net.serenitybdd.screenplay.actors.OnStage;
12 import net.serenitybdd.screenplay.actors.OnlineCast;
13
14 public class ChoucairAcademyStepDefintions {
15     @Before
16     public void setStage() { OnStage.setTheStage(new OnlineCast()); }
17
18     @Given("^than brandon wants to learn automation at the academy Choucair$")
19     public void thanBrandonWantsToLearnAutomationAtTheAcademyChoucair() {
20         OnStage.theActorCalled( requiredActor: "Brandon").wasAbleTo(OpenUp.thePage(), (Login.onThePage()));
21     }
22
23     @When("^he search for the course (.*) on the choucair academy platform$")
24     public void heSearchForTheCourseRecursosAutomatizaciónBancolombiaOnTheChoucairAcademyPlatform(String course) {
25         OnStage.theActorInTheSpotLight().attemptsTo(Search.the(course));
26     }
27
28     @Then("^he finds the course called resources Recursos Automatización Bancolombia$")
29     public void heFindsTheCourseCalledResourcesRecursosAutomatizaciónBancolombia() {
30         OnStage.theActorInTheSpotLight().should(GivenWhenThen.seeThat(Answer.toThe(question)));
31     }
32 }
```

En el método seleccionado dentro del cuadro rojo, es donde estaremos trabajando la implementación para nuestra verificación en el “Question”. Para convertirlo en algo más dinámico, usaremos como en la guía pasada las expresiones regulares pertinentes.

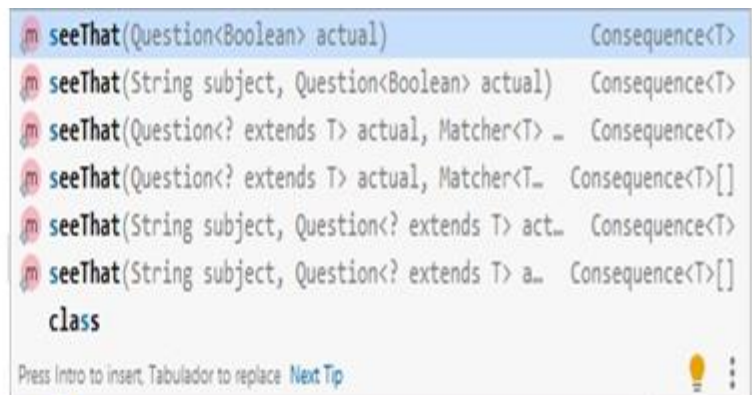
Reemplazando la palabra “Recursos Automatización Bancolombia” por “(.*)” y declarando un parámetro String, tendremos lo siguiente:

```
@Then("^he finds the course called resources (.*)$")
public void heFindsTheCourseCalledResourcesRecursosAutomatizaciónBancolombia(String question) {
```

Para escribir Questions, usaremos el método “**should**” de nuestro Actor.

```
OnStage.theActorInTheSpotLight().should();
```

Dentro escribiremos entonces **GivenWhenThen** seguido de un punto y escogeremos el método **seeThat**.



```
OnStage.theActorInTheSpotLight().should(GivenWhenThen.s);
```

Obtendremos una línea igual a la siguiente:

```
@Then("^he finds the course called resources (.*)$")
public void heFindsTheCourseCalledResourcesRecursosAutomatizaciónBancolombia(String question) {
    OnStage.theActorInTheSpotLight().should(GivenWhenThen.seeThat(Answer));
}
```

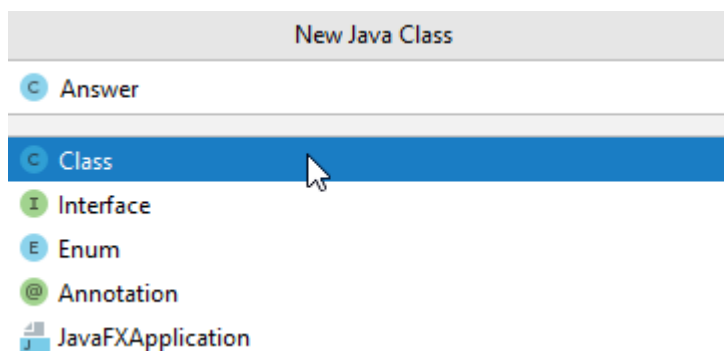
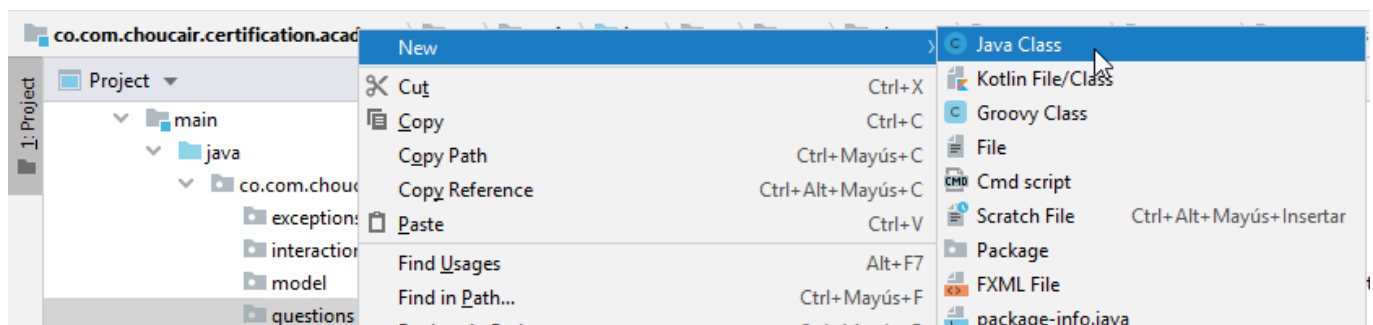
Donde en “Answer” escribiremos nuestra pregunta. Para escribir la pregunta, seguiremos la misma lógica de una tarea, por lo tanto, obtenemos:



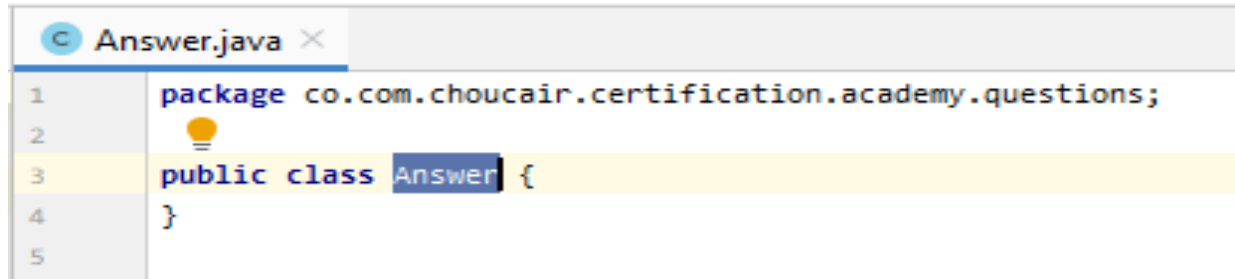
```
@Then("^he finds the course called resources (.*)$")
public void heFindsTheCourseCalledResourcesRecursosAutomatizaciónBancolombia(String question) {
    OnStage.theActorInTheSpotlight().should(GivenWhenThen.seeThat(Answer.toThe(question)));
}
```

Donde “toThe” será un método de clase Answer que crearemos en nuestro paquete “src/main/java/co.com.choucair.certification.academy” dentro del paquete “questions”.

Ahora procedemos a crear nuestra **Question** en el paquete que corresponde: Para ello nos posicionaremos en el paquete “questions”, daremos clic derecho en New y damos clic en Java Class y crearemos la clase Answer.

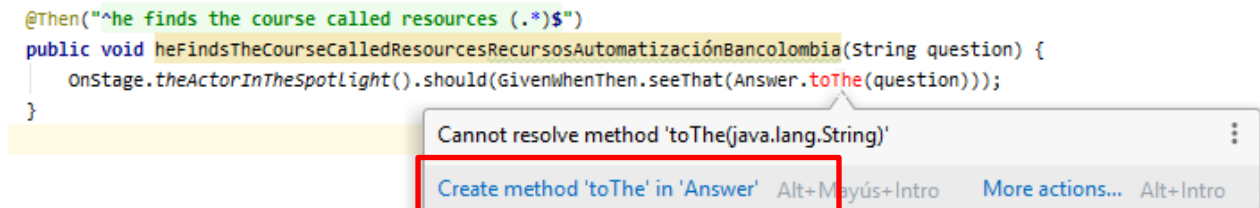


Verificamos que se haya creado una clase llamada Answer, la cual está completamente vacía.



```
1 package co.com.choucair.certification.academy.questions;
2
3 public class Answer {
4 }
5
```

Creamos también el método estático en esta clase:

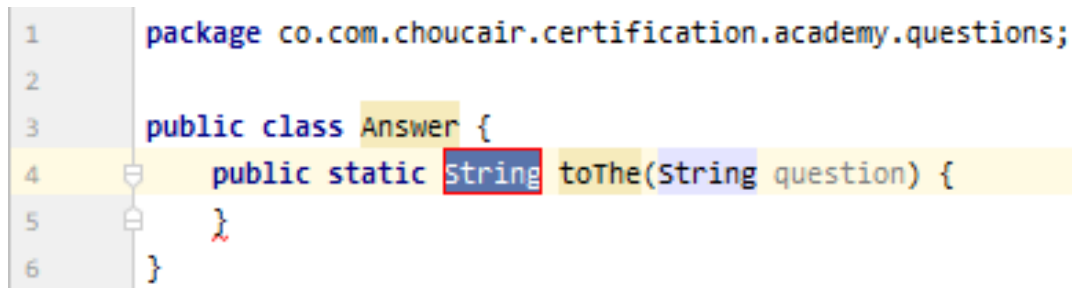


```
@Then("^he finds the course called resources (.*)$")
public void heFindsTheCourseCalledResourcesRecursosAutomatizaciónBancolombia(String question) {
    OnStage.theActorInTheSpotlight().should(GivenWhenThen.seeThat(Answer.toThe(question)));
}
```

Cannot resolve method 'toThe(java.lang.String)'

Create method 'toThe' in 'Answer' Alt+Mayús+Intro More actions... Alt+Intro

Y tendremos nuestra clase de la siguiente forma:



```
1 package co.com.choucair.certification.academy.questions;
2
3 public class Answer {
4     public static String toThe(String question) {
5     }
6 }
```



Para los Questions usaremos una interface llamada Question y como si fuese del tipo List, le definiremos el tipo que deseamos tener como respuesta, int, String, doublé, etc... En este caso, será del tipo Boolean.

```
import net.serenitybdd.screenplay.Question;

public class Answer implements Question<Boolean> {
    public static String toThe(String question) {
    }
}
```

Y agregamos los métodos que no han sido implementados aun:

```
1 package co.com.choucair.certification.academy.questions;
2
3 import net.serenitybdd.screenplay.Question;
4
5 public class Answer implements Question<Boolean> {
6     public static String toThe(String question) {
7         return null;
8     }
9 }
10
```

Class 'Answer' must either be declared abstract or implement abstract method 'answeredBy(Actor)' in 'Question'

Implement methods Alt+Mayús+Intro More actions... Alt+Intro

Nuestra clase debería ir así:

```
public class Answer implements Question<Boolean> {
    private String question;

    public Answer(String question) {
        this.question = question;
    }

    public static Answer toThe(String question) {
        return new Answer(question);
    }

    @Override
    public Boolean answeredBy(Actor actor) {
        return null;
    }
}
```



Ahora procedemos a escribir la línea que tomará el texto del título del curso, para realizar este paso inspeccionaremos el elemento igual que en las guías anteriores y se lo asignaremos a una variable de tipo String. En cuanto a las Questions, toda “la magia” la haremos en el método llamado “answeredBy”. Para obtener el String usaremos la sentencia “Text.of”, le pasaremos nuestro objeto tipo Target, y le diremos que es visto por nuestro actor como un String.

```
String nameCourse= Text.of(SearchCoursePage.NAME_COURSE).viewedBy(actor).asString();
```

NAME_COURSE, se mapeo el objeto en la clase SearchCoursePage.

```
public static final Target NAME_COURSE = Target.the( targetElementName: "Extrae el nombre del curso")  
    .located(By.xpath("//h1[contains(text(),'Recursos Automatización Bancolombia')]"));
```

Dado que la validación que vamos a hacer es de tipo booleano, realizaremos una condición con un “if” donde se evaluara que el contenido del parámetro “question” sea igual al dato que tenemos en nuestra variable “nameCourse”.

```
@Override  
public Boolean answeredBy(Actor actor) {  
    boolean result;  
    String nameCourse= Text.of(SearchCoursePage.NAME_COURSE).viewedBy(actor).asString();  
    if (question.equals(nameCourse)){  
        result = true;  
    }else {  
        result = false;  
    }  
    return result;  
}
```



Nuestra clase quedaría finalmente así:

```
Answer.java x
1 package co.com.choucair.certification.academy.questions;
2
3 import co.com.choucair.certification.academy.userinterface.SearchCoursePage;
4 import net.serenitybdd.screenplay.Actor;
5 import net.serenitybdd.screenplay.Question;
6 import net.serenitybdd.screenplay.questions.Text;
7
8 public class Answer implements Question<Boolean> {
9     private String question;
10
11     @ public Answer(String question) {
12         this.question = question;
13     }
14
15     @ public static Answer toThe(String question) {
16         return new Answer(question);
17     }
18
19     @Override
20     public Boolean answeredBy(Actor actor) {
21         boolean result;
22         String nameCourse= Text.of(SearchCoursePage.NAME_COURSE).viewedBy(actor).asString();
23         if (question.equals(nameCourse)){
24             result = true;
25         }else {
26             result = false;
27         }
28         return result;
29     }
30 }
31
```

Answer > answeredBy()



y nuestro stepDefinitions quedara así:

```
ChoucairAcademyStepDefintions.java X
1 package co.com.choucair.certification.academy.stepdefinitions;
2
3 import co.com.choucair.certification.academy.questions.Answer;
4 import co.com.choucair.certification.academy.tasks.Login;
5 import co.com.choucair.certification.academy.tasks.OpenUp;
6 import co.com.choucair.certification.academy.tasks.Search;
7 import cucumber.api.java.Before;
8 import cucumber.api.java.en.Given;
9 import cucumber.api.java.en.Then;
10 import cucumber.api.java.en.When;
11 import net.serenitybdd.screenplay.GivenWhenThen;
12 import net.serenitybdd.screenplay.actors.OnStage;
13 import net.serenitybdd.screenplay.actors.OnlineCast;
14
15 public class ChoucairAcademyStepDefintions {
16     @Before
17     public void setStage() { OnStage.setTheStage(new OnlineCast()); }
18
19     @Given("^than brandon wants to learn automation at the academy Choucair$")
20     public void thanBrandonWantsToLearnAutomationAtTheAcademyChoucair() {
21         OnStage.theActorCalled( requiredActor: "Brandon").wasAbleTo(OpenUp.thePage(), (Login.onThePage()));
22     }
23
24     @When("^he search for the course (.*) on the choucair academy platform$")
25     public void heSearchForTheCourseRecursosAutomatizaciónBancolombiaOnTheChoucairAcademyPlatform(String course) {
26         OnStage.theActorInTheSpotlight().attemptsTo(Search.the(course));
27     }
28
29     @Then("^he finds the course called resources (.*)$")
30     public void heFindsTheCourseCalledResourcesRecursosAutomatizaciónBancolombia(String question) {
31         OnStage.theActorInTheSpotlight().should(GivenWhenThen.seeThat(Answer.toThe(question)));
32     }
33 }
34
35
36
37
38
```



Al correr nuestra prueba, y de haber seguido correctamente las guías, veremos que todo ha salido perfecto.

✓ Tests passed: 1 of 1 test – 25 s 121 ms

[illegible]

TEST PASSED: Search for a automation course

```
[Test worker] INFO net.serenitybdd.core.Serenity -
```

[illegible]

TEST PASSED: Search for a automation course

1 Scenarios (1 passed)

3 Steps (3 passed)
0m25,981s

```
BUILD SUCCESSFUL in 35s
5 actionable tasks: 4 executed, 1 up-to-date
```

```
11:21:39 a. m.: Tasks execution finished ':cleanTest :test --tests "co.com.choucair.certification.academy.runners.RunnerTags"'.

```

¡Felicidades ya conoces la implementación básica

¡Felicidades ya conoces la implementación básica para el patrón Screenplay! ¡Ahora solo te falta ampliar tus conocimientos con estudio y práctica!

