

ScreenPlay +BDD+Appium+Android 4 - Implementación de questions

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

Contenido

| | |
|--|---|
| Implementación del Then..... | 1 |
| Verificamos la ejecución por comandos de gradle..... | 8 |

Implementación del Then

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 you should see the login in the application with the message Logged in as



ScreenPlay +BDD+Appium+Android 4 - Implementación de questions

Entonces iremos nuevamente a nuestra clase “AuthenticationStepDefinitions”, la cual debe lucir de la siguiente forma después de las dos tareas implementadas.

```
AuthenticationStepDefinitions.java X
1 package co.com.choucair.automation.android.stepdefinitions;
2
3 import co.com.choucair.automation.android.tasks.Login;
4 import co.com.choucair.automation.android.tasks.OpenThe;
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.actors.OnStage;
9
10 public class AuthenticationStepDefinitions {
11     @Given("^that (.*) wants to enter the Wordpress application$")
12     public void thatBrandonWantsToEnterTheWordpressApplication(String brandon) {
13         OnStage.theActorCalled(brandon).wasAbleTo(OpenThe.wordpressApp());
14     }
15
16     @When("^I login with the username \"([^\"]*)\" and the password \"([^\"]*)\"$")
17     public void iLoginWithTheUsernameAndThePassword(String user, String password) {
18         OnStage.theActorInTheSpotlight().attemptsTo(Login.with(user,password));
19     }
20
21     @Then("^you should see the login in the application with the message Logged in as$")
22     public void youShouldSeeTheLoginInTheApplicationWithTheMessageLoggedInAs() {
23     }
24 }
25
```

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, entonces usaremos como en la guía pasada las expresiones regulares pertinentes. Reemplazando la palabra “Logged in as” por “(.*?)” y declarando un parámetro String, tendremos lo siguiente:

ScreenPlay +BDD+Appium+Android 4 - Implementación de questions

```
@Then("^you should see the login in the application with the message (.*)$")
public void youShouldSeeTheLoginInTheApplicationWithTheMessageLoggedInAs(String question) {
}
```

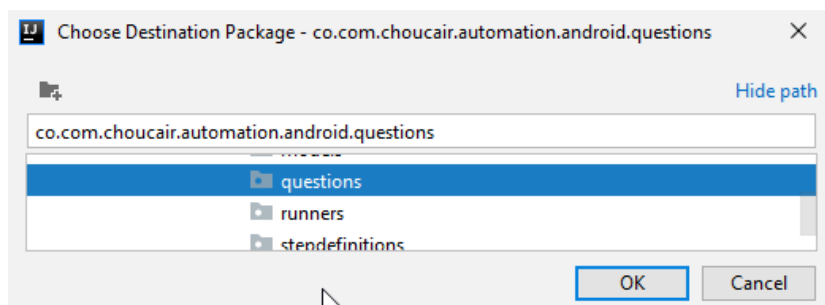
Para escribir la questions, usaremos el método “should” de nuestro Actor. Dentro escribiremos entonces GivenWhenThen seguido de un punto y escogeremos el método seeThat. Y dentro de seeThat escribiremos “**VerifyWith**” será una clase que crearemos en nuestro paquete “**questions**” Y el método “**the**”, será un método estático de dicha clase, a la cual le pasaremos por parámetro la variable “**question**”.

```
@Then("^you should see the login in the application with the message (.*)$")
public void youShouldSeeTheLoginInTheApplicationWithTheMessageLoggedInAs(String question) {
    OnStage.theActorInTheSpotLight().should(GivenWhenThen.seeThat(VerifyWith.the(question)));
}
```

Ahora procedemos a crear nuestra validación en el paquete que corresponde:

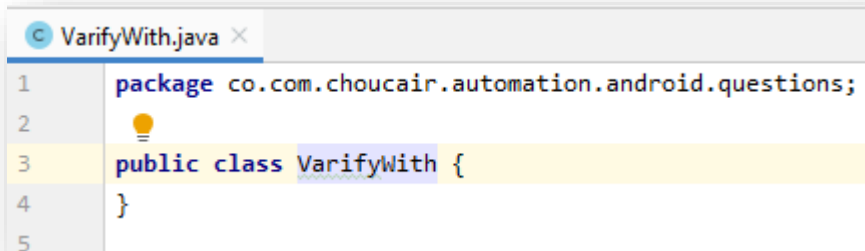
```
@Then("^you should see the login in the application with the message (.*)$")
public void youShouldSeeTheLoginInTheApplicationWithTheMessageLoggedInAs(String question) {
    OnStage.theActorInTheSpotLight().should(GivenWhenThen.seeThat(VerifyWith.the(question)));
}
```

- Create class 'VerifyWith'
- Create field 'VerifyWith' in 'AuthenticationStepDefinitions'
- Create inner class 'VerifyWith'
- Create local variable 'VerifyWith'
- Create parameter 'VerifyWith'
- Rename reference



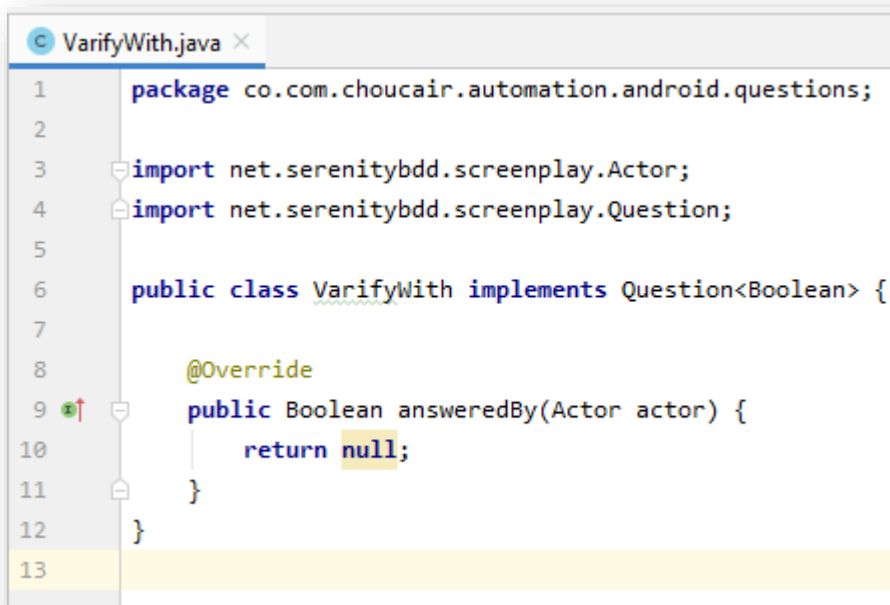
ScreenPlay +BDD+Appium+Android 4 - Implementación de questions

Verificamos que se haya creado una clase llamada **VerifyWith**, y está completamente vacía.



```
1 package co.com.choucair.automation.android.questions;
2
3 public class VerifyWith {
4 }
5
```

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. Y agregamos los métodos que no han sido implementados aun:



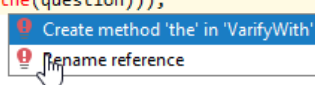
```
1 package co.com.choucair.automation.android.questions;
2
3 import net.serenitybdd.screenplay.Actor;
4 import net.serenitybdd.screenplay.Question;
5
6 public class VerifyWith implements Question<Boolean> {
7
8     @Override
9     public Boolean answeredBy(Actor actor) {
10         return null;
11     }
12 }
13
```



ScreenPlay +BDD+Appium+Android 4 - Implementación de questions

Ahora volvemos a nuestra clase stepdefinitions y creamos el método “the”.

```
@Then("^you should see the login in the application with the message (.*)$")
public void youShouldSeeTheLoginInTheApplicationWithTheMessageLoggedInAs(String question) {
    OnStage.theActorInTheSpotLight().should(GivenWhenThen.seeThat(VerifyWith.the(question)));
}
```

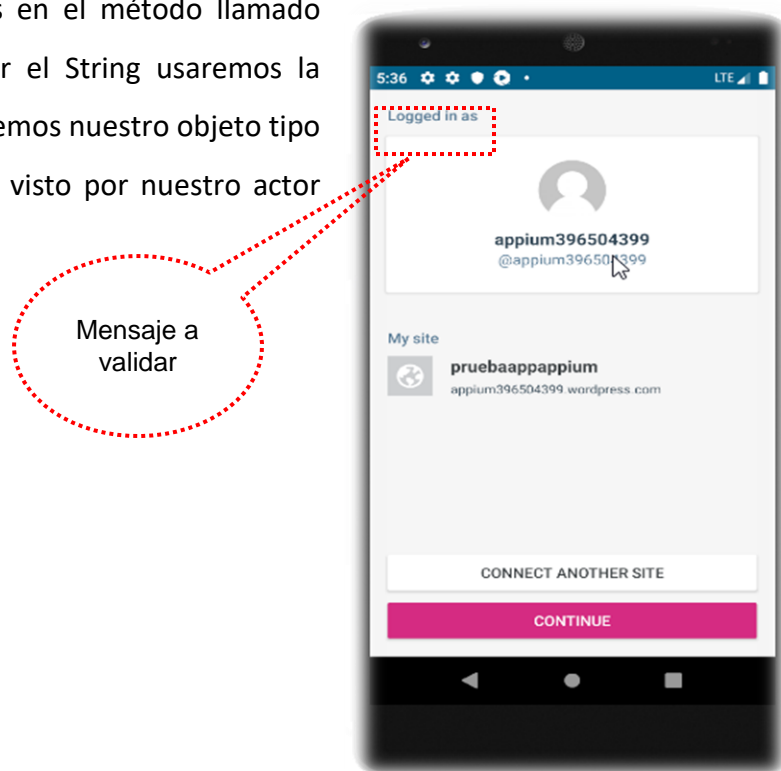


Automáticamente nos creara el método **the** en la clase **VerifyWith** , el debe retornar el parámetro question de la clase y para eso es necesario crear un constructor a la variable question que también declararemos de tipo private. Tu clase se debe ver así:

```
VarifyWith.java x
1 package co.com.choucair.automation.android.questions;
2
3 import net.serenitybdd.screenplay.Actor;
4 import net.serenitybdd.screenplay.Question;
5
6 public class VarifyWith implements Question<Boolean> {
7
8     private String question;
9
10    public VarifyWith(String question) {
11        this.question = question;
12    }
13
14    @ public static VarifyWith the(String question) {
15        return new VarifyWith(question);
16    }
17
18    @Override
19    public Boolean answeredBy(Actor actor) {
20        return null;
21    }
22 }
```

ScreenPlay +BDD+Appium+Android 4 - Implementación de questions

Ahora procedemos a escribir la línea que tomará de la app el “Logged in as”, desde el label de la app de wordpress, objeto inspeccionado en la guía anterior al que llamamos **TEXT_LABEL** 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.



El mapeo de le objeto quedo así: en la clase **LoginPage**.

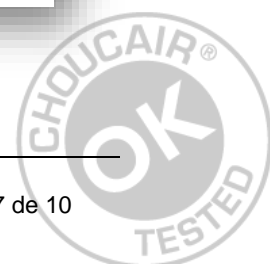
```
public static final Target TEXT_LABEL = Target.the("text label for validations")
    .located(By.id("org.wordpress.android:id/logged_in_as_heading"));
```

ScreenPlay +BDD+Appium+Android 4 - Implementación de questions

Nuestra clase VerifyWith y la clase AuthenticationStepDefinitions quedarían finalmente así:

```
VerifyWith.java
1 package co.com.choucair.automation.android.questions;
2
3 import net.serenitybdd.screenplay.Actor;
4 import net.serenitybdd.screenplay.Question;
5 import net.serenitybdd.screenplay.questions.Text;
6
7 import static co.com.choucair.automation.android.userinterfaces.LoginPage.TEXT_LABEL;
8
9 public class VerifyWith implements Question<Boolean> {
10
11     private String question;
12
13     public VerifyWith(String question) {
14         this.question = question;
15     }
16
17     @Override
18     public static VerifyWith the(String question) {
19         return new VerifyWith(question);
20     }
21
22     @Override
23     public Boolean answeredBy(Actor actor) {
24         String answer = Text.of(TEXT_LABEL).viewedBy(actor).asString();
25         return question.equals(answer);
26     }
27 }

AuthenticationStepDefinitions.java
1 package co.com.choucair.automation.android.stepdefinitions;
2
3 import ...
4
5
6
7
8
9
10
11
12 public class AuthenticationStepDefinitions {
13     @Given("^that (.*) wants to enter the Wordpress application$")
14     public void thatBrandonWantsToEnterTheWordpressApplication(String brandon) {
15         OnStage.theActorCalled(brandon).wasAbleTo(OpenThe.wordpressApp());
16     }
17
18     @When("^I login with the username \"([^\"]*)\" and the password \"([^\"]*)\"$")
19     public void iLoginWithTheUsernameAndThePassword(String user, String password) {
20         OnStage.theActorInTheSpotLight().attemptsTo(Login.with(user, password));
21     }
22
23     @Then("^you should see the login in the application with the message (.*)$")
24     public void youShouldSeeTheLoginInTheApplicationWithTheMessageLoggedInAs(String question) {
25         OnStage.theActorInTheSpotLight().should(GivenWhenThen.seeThat(VerifyWith.the(question)));
26     }
27 }
28 }
```

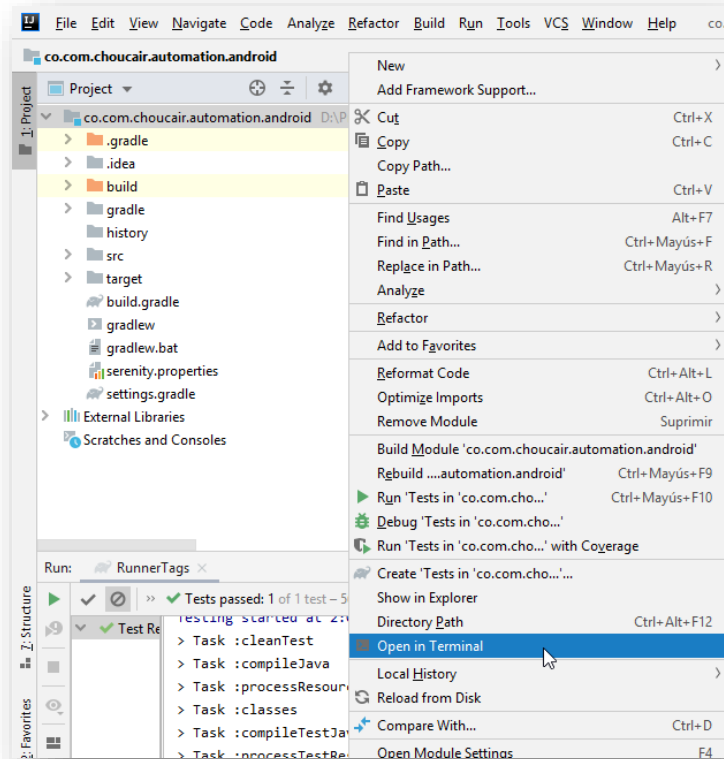


ScreenPlay +BDD+Appium+Android 4 - Implementación de questions

Verificamos la ejecución por comandos de gradle

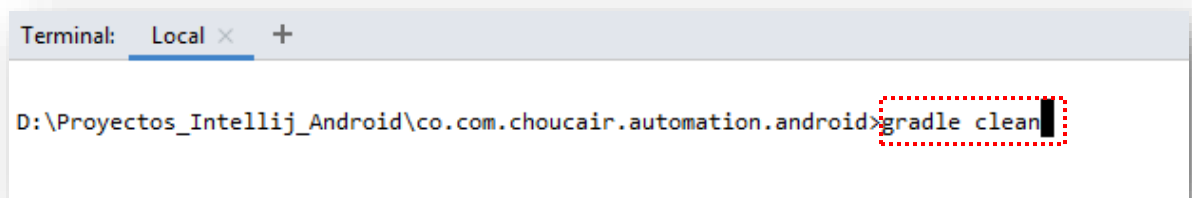
Haciendo uso de la consola vamos a ejecutar nuestra automatización por líneas de comando de gradle.

Para ello vamos a dar clic derecho sobre el proyecto y vamos a escoger la opción "Open in Terminal"



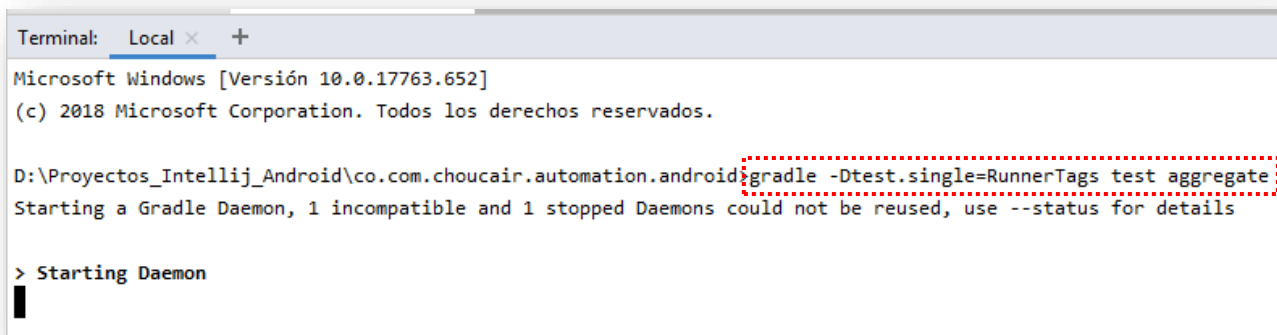
Seguido en la consola vamos a usar los siguientes comandos:

1. Ejecutamos el comando: **gradle clean**.



ScreenPlay +BDD+Appium+Android 4 - Implementación de questions

- luego ejecutamos el comando: **gradle -Dtest.single=RunnerTags test aggregate**

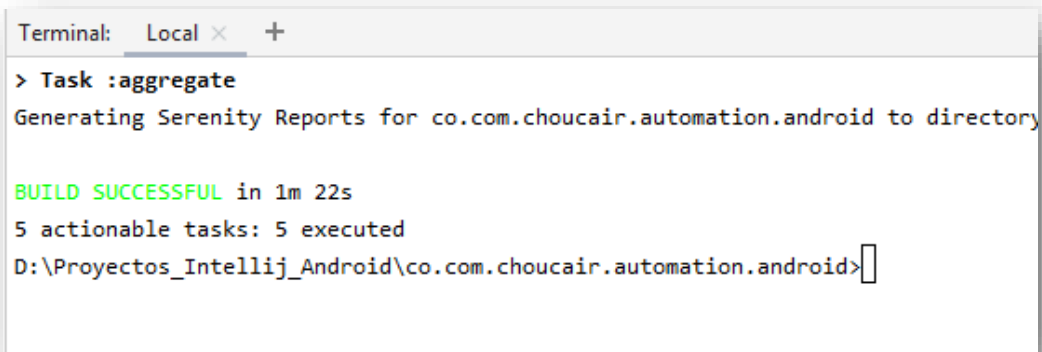


```
Terminal: Local x +
Microsoft Windows [Versión 10.0.17763.652]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

D:\Proyectos_Intellij_Android\co.com.choucair.automation.android>gradle -Dtest.single=RunnerTags test aggregate
Starting a Gradle Daemon, 1 incompatible and 1 stopped Daemons could not be reused, use --status for details

> Starting Daemon
█
```

- terminada la ejecución



```
Terminal: Local x +

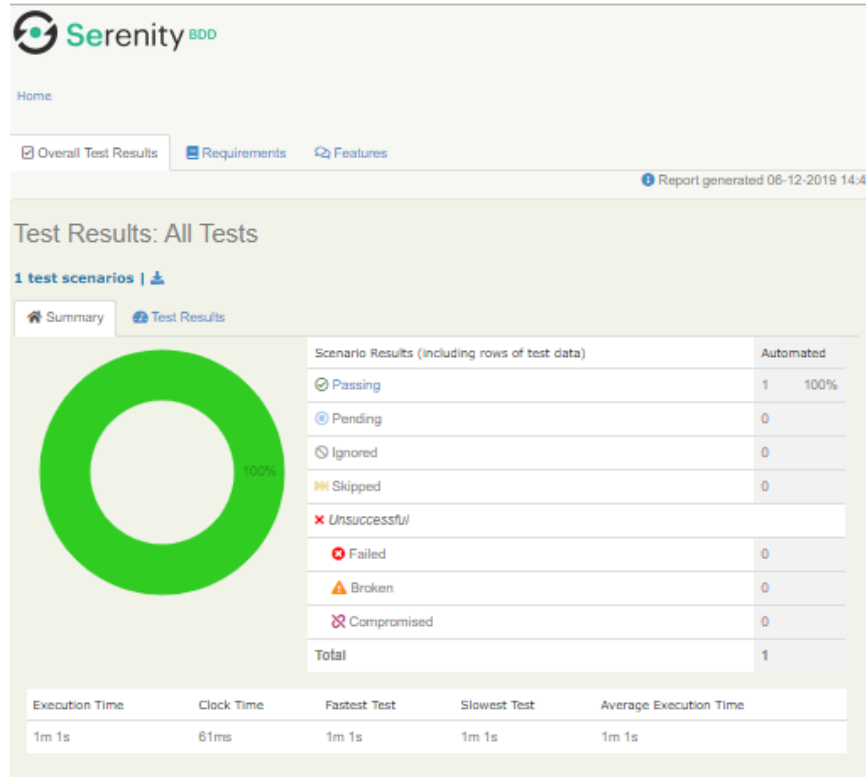
> Task :aggregate
Generating Serenity Reports for co.com.choucair.automation.android to directory

BUILD SUCCESSFUL in 1m 22s
5 actionable tasks: 5 executed
D:\Proyectos_Intellij_Android\co.com.choucair.automation.android>
```

- Una vez generado el reporte damos clic derecho al folder: **/target/site/serenity**
clic derecho > Dar clic al botón “Show in system explorer” ingresamos a la carpeta
sereniti y abrimos el archivo que se llama index.html con un navegador



ScreenPlay +BDD+Appium+Android 4 - Implementación de questions



**¡Felicidades ya conoces la implementación
básica para el patrón Screenplay usando
APPIUM!**

