

ScreenPlay +BDD+WinAppDriver 2 - Configuración Inicial, Feature y StepDefinitons

Configuración inicial, feature y StepDefinition

En la presente guía, daremos nuestros primeros pasos usando el patrón Screenplay con WinAppDriver para automatización de escritorio. Una vez cargado nuestro proyecto base en intellij, procederemos a crear la Configuración Inicial, Feature y StepDefinitions.

Tabla de contenido

Importar proyecto Base	2
Mi primera prueba usando Screenplay BDD con WinAppDriver	4
Creación del feature.	6
Creación del stepdefinitions.	8

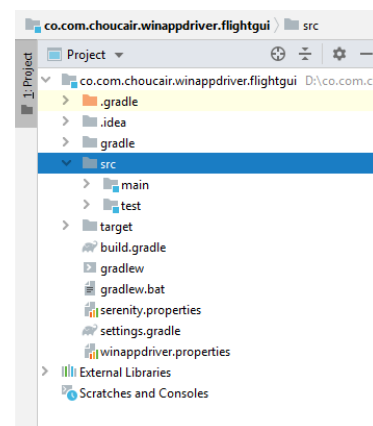
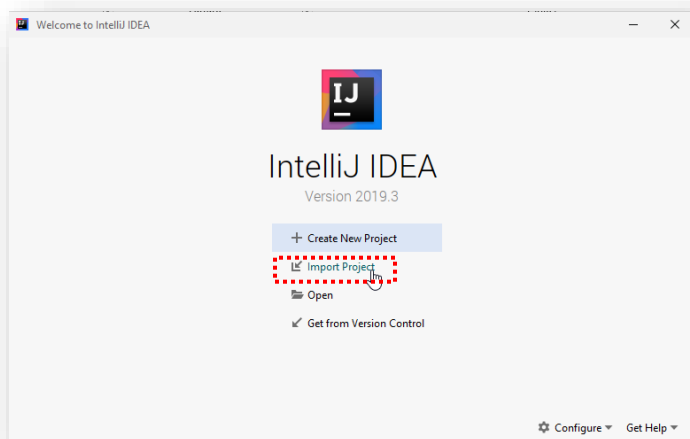


ScreenPlay +BDD+WinAppDriver 2 - Configuración Inicial, Feature y StepDefinitons

Importar proyecto Base

Se cuenta con un proyecto base el cual contiene la estructura del patrón de screenplay para automatización de aplicaciones de escritorio.

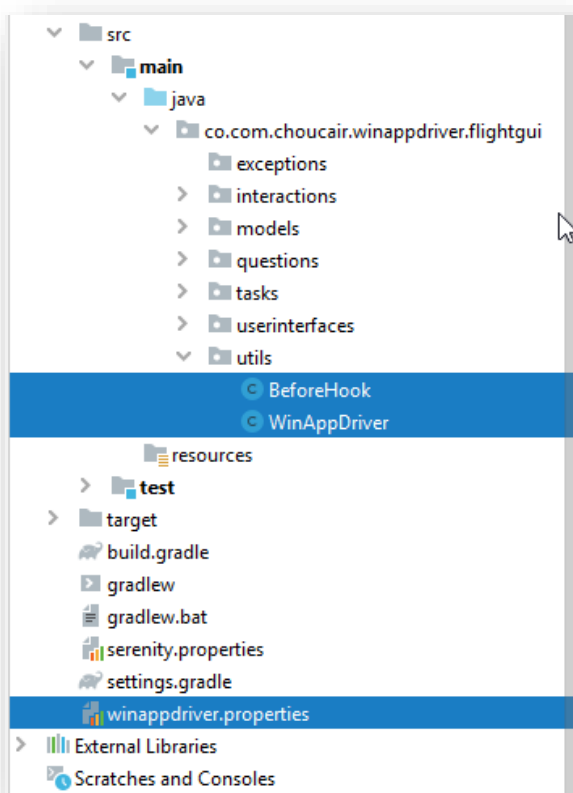
1. Descarga el proyecto base en la ruta del workspace configurado del siguiente link:
[Proyecto Base WinAppDriver](#).
2. Descomprime el proyecto en la ruta de tu elección.
3. Importar el proyecto.
 - a. Iniciamos abriendo IntelliJ IDEA y damos clic en Import Project



- b. Seleccionamos la ruta donde descomprimos el proyecto y damos clic en OK.
- c. Escoge la opción de importar proyecto y que además sea de tipo Gradle y da clic en Finish.
- d. Este proceso puede durar varios minutos, permita que el proceso finalice por completo y debe quedar como la segunda imagen.

ScreenPlay +BDD+WinAppDriver 2 - Configuración Inicial, Feature y StepDefinitons

En el proyecto base, en el paquete “*utils*” se encuentran dos clases **WinAppDriver** y **BeforeHook**, la primera clase se encarga de abrir automáticamente el ejecutable del servicio de WinAppDriver.exe y de cerrarlo si se encuentra corriendo. En la segunda clase se hace el llamado de la primera clase y además se prepara el escenario para el actor. Además, en la raíz del proyecto hay un archivo llamado “*winappdriver.properties*” el cual contiene la ruta del ejecutable del **WinAppDriver.exe**.



Nota: Si borras alguno de estos archivos (*WinAppDriver* o *winappdriver.properties*) es necesario que manualmente ejecute windows application driver antes de ejecutar la automatización.

ScreenPlay +BDD+WinAppDriver 2 - Configuración Inicial, Feature y StepDefinitons

Mi primera prueba usando Screenplay BDD con WinAppDriver



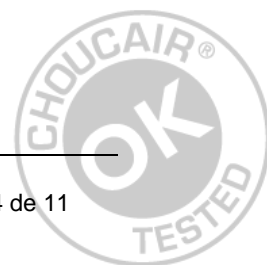
Flight GUI
Aplicación

Historia de Usuario: comprar un boleto aéreo en la aplicación de escritorio flight GUI, en ella vamos a loguearnos, ingresaremos la información para comprar un ticket y verificamos con un mensaje.

Criterios de Aceptación: Verificar la compra de un boleto aéreo

Pasos para la ejecución de la prueba.

1. compra de un boleto aéreo
 - a. Abrir la aplicación de escritorio flight GUI.
 - b. Ingresar "john" en el campo Username
 - c. Ingresar "HP" en el campo Password
 - d. Dar clic en el botón OK
 - e. En el apartado de BOOK FLIGHT diligenciar todos los campos (origen, destino, fecha, tipo de vuelo, cantidad de tickets).
 - f. Dar clic en el botón FIND FLIGHT
 - g. Seleccionar un vuelo de la tabla
 - h. Dar clic en el botón SELECT FLIGHT
 - i. En el apartado FLIGHT DETAILS diligencia el campo(Passenger Name).
 - j. Dar clic en el botón ORDER
 - k. Verificar el registro validando que contenga la palabra "completed"



ScreenPlay +BDD+WinAppDriver 2 - Configuración Inicial, Feature y StepDefinitons

Análisis:

Existen diferentes formas de analizar el flujo de una transacción, para efectos de este taller vamos a agrupar los pasos en acciones concretas y específicas. Dichas acciones serán escritas en lenguaje Gherkin en nuestra feature.

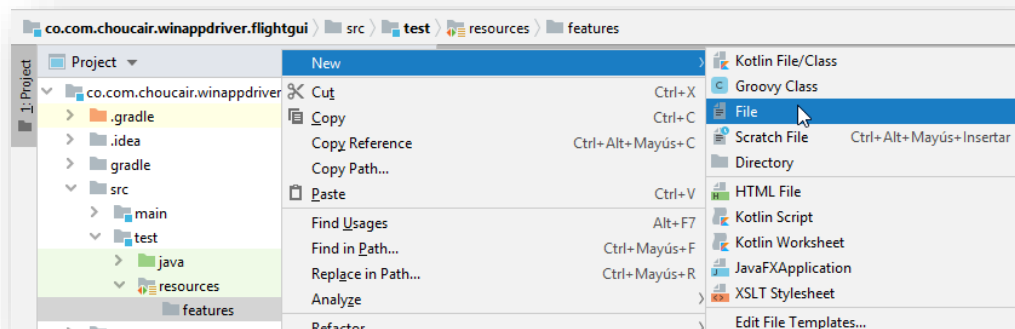
Como siempre en cualquier prueba es importante que determinemos la data requerida para la ejecución de la prueba.



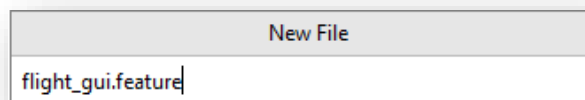
ScreenPlay +BDD+WinAppDriver 2 - Configuración Inicial, Feature y StepDefinitons

Creación del feature.

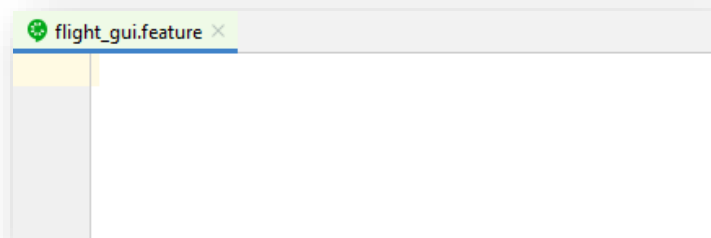
1. Ingresamos a la ruta **src/test/resources/features**
2. Sobre la capeta feature damos clic derecho > New > File



3. Y le ponemos el nombre al archivo para este ejemplo lo llamaremos **"flight_gui.feature"** y presionamos la tecla **Enter**.



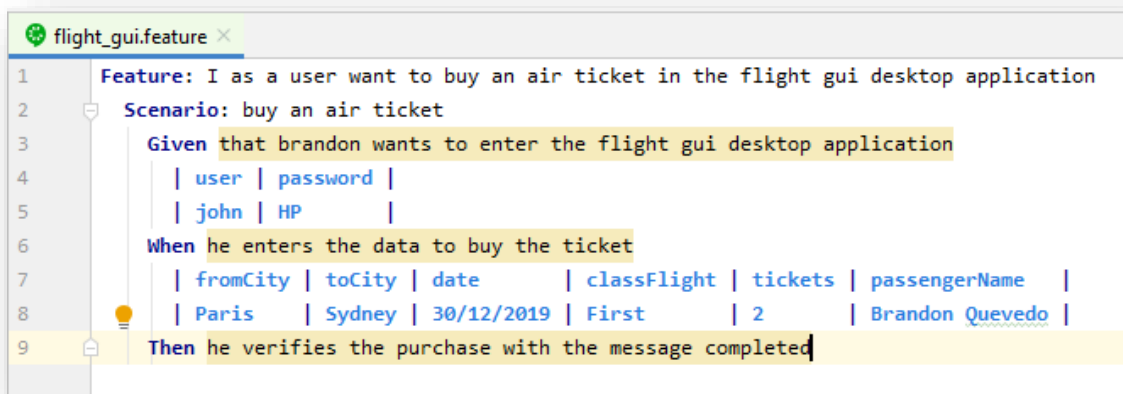
4. Nos debe generar un archivo como el de la siguiente imagen en donde redactaremos nuestra historia de usuario



ScreenPlay +BDD+WinAppDriver 2 - Configuración Inicial, Feature y StepDefinitons

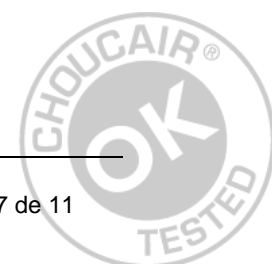
5. A continuación, redactamos nuestra historia de usuario en el archivo "flight_gui.feature"

Nota: debido a que el idioma nativo de Cucumber es Ingles, de aquí en adelante tanto nuestra historia de usuario (feature) y nuestra automatización la realizaremos en ese idioma.



```
1 Feature: I as a user want to buy an air ticket in the flight gui desktop application
2 Scenario: buy an air ticket
3   Given that brandon wants to enter the flight gui desktop application
4       | user | password |
5       | john | HP      |
6   When he enters the data to buy the ticket
7       | fromCity | toCity | date      | classFlight | tickets | passengerName |
8       | Paris   | Sydney | 30/12/2019 | First       | 2       | Brandon Quevedo |
9   Then he verifies the purchase with the message completed
```

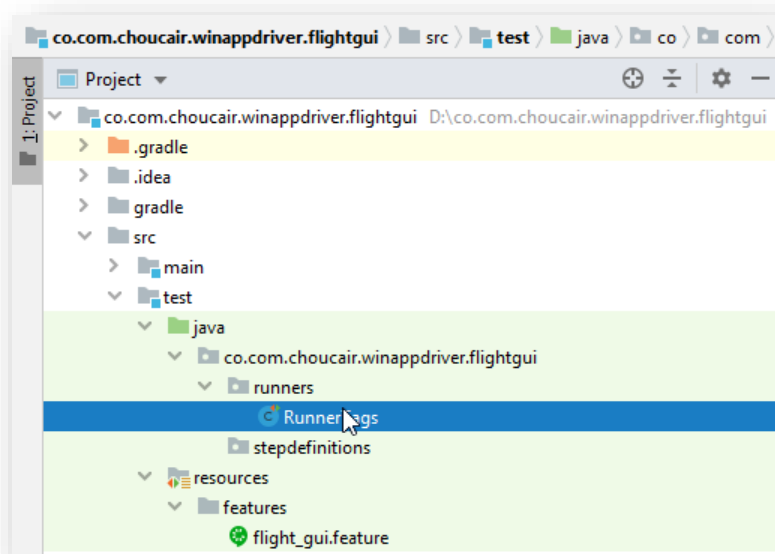
Una vez redactada nuestra historia de usuario, procedemos a ejecutar la misma con nuestro Runner para poder crear nuestra clase stepdefinition y colocar en esta los métodos que deben ser implementados.



ScreenPlay +BDD+WinAppDriver 2 - Configuración Inicial, Feature y StepDefinitons

Creación del stepdefinitions.

Para crear nuestro StepDefinitions vas a ir al proyecto base en intellij a la ruta **src > test > java > co.com.choucair.winappdriver.flightgui > runners** y abre el archivo llamado **RunnerTags**.



ScreenPlay +BDD+WinAppDriver 2 - Configuración Inicial, Feature y StepDefinitons

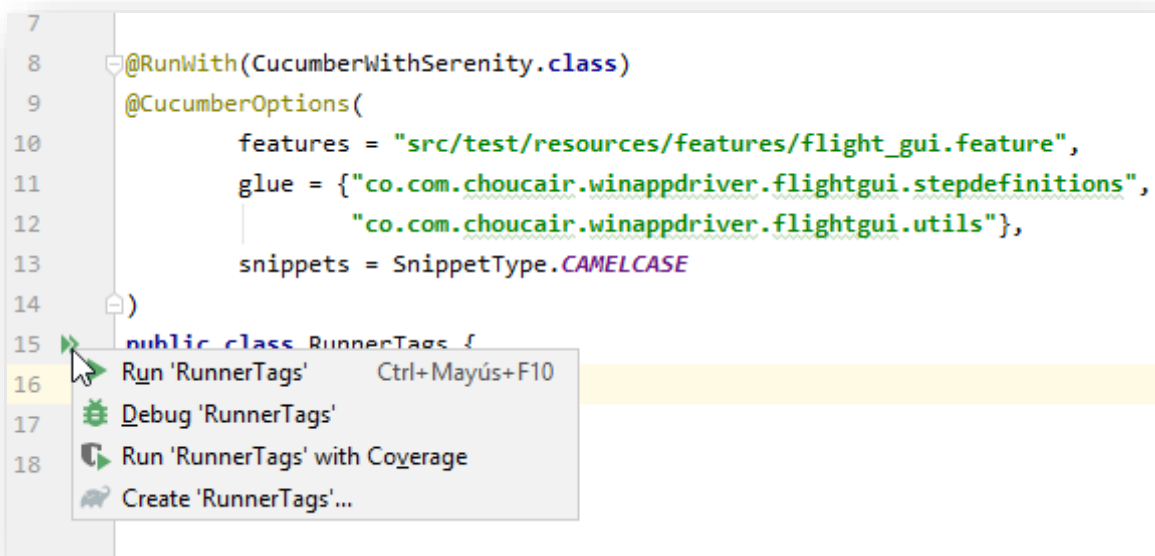
```
1 package co.com.choucair.winappdriver.flightgui.runners;
2
3 import cucumber.api.CucumberOptions;
4 import cucumber.api.SnippetType;
5 import net.serenitybdd.cucumber.CucumberWithSerenity;
6 import org.junit.runner.RunWith;
7
8 @RunWith(CucumberWithSerenity.class)
9 @CucumberOptions(
10     features = "src/test/resources/features/flight_gui.feature",
11     glue = {"co.com.choucair.winappdriver.flightgui.stepdefinitions",
12           "co.com.choucair.winappdriver.flightgui.utils"},
13     snippets = SnippetType.CAMEL_CASE
14 )
15 public class RunnerTags {
16 }
17
```

El runner será donde se iniciará la ejecución de nuestra prueba, y también será el encargado de compilar la feature para generar los métodos a implementar en el stepdefinitions.

Para ejecutar la clase RunnerTags, clic derecho

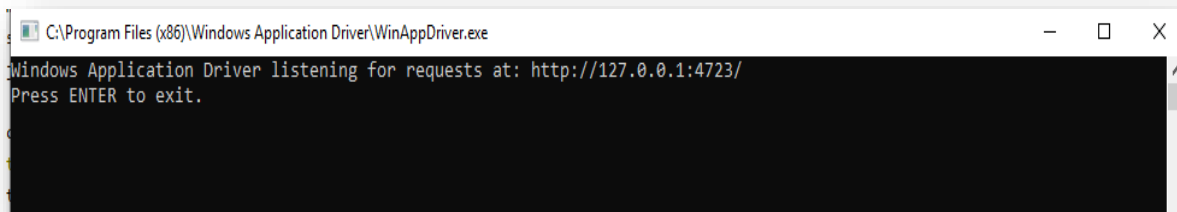
sobre el runner > Run 'RunnerTags'.

```
7
8 @RunWith(CucumberWithSerenity.class)
9 @CucumberOptions(
10     features = "src/test/resources/features/flight_gui.feature",
11     glue = {"co.com.choucair.winappdriver.flightgui.stepdefinitions",
12           "co.com.choucair.winappdriver.flightgui.utils"},
13     snippets = SnippetType.CAMEL_CASE
14 )
15 public class RunnerTags {
16
17
18
```

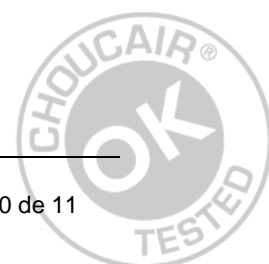
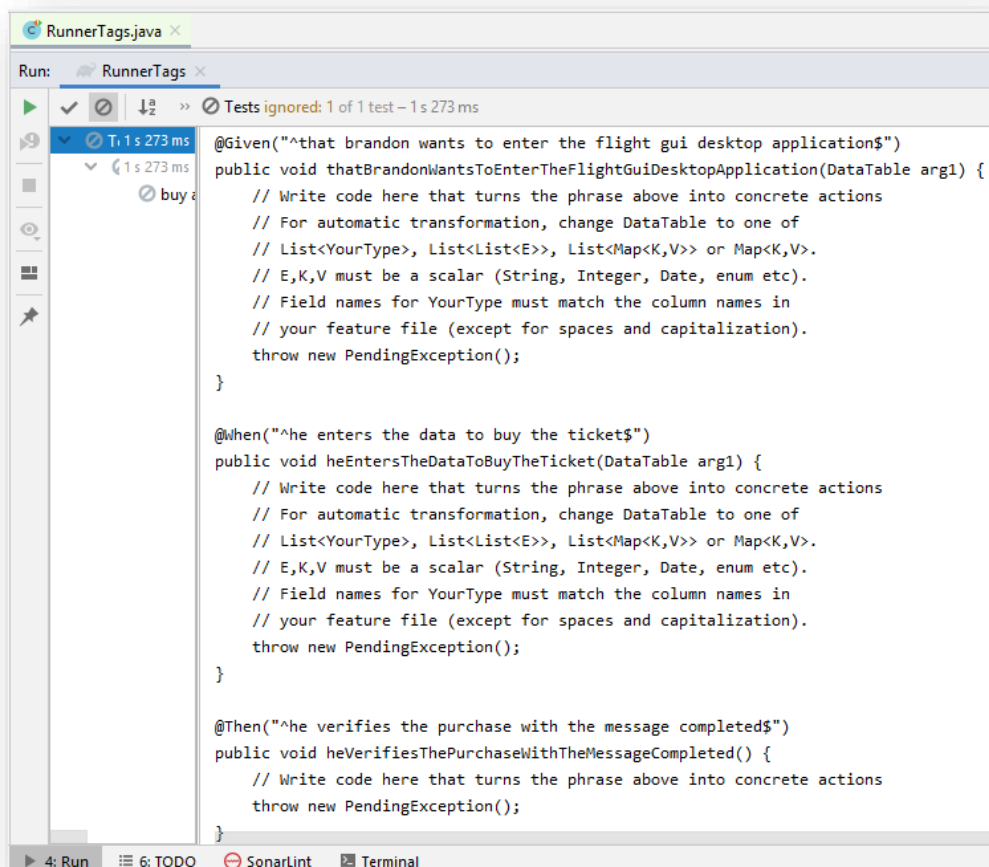


Se abrirá la ventana del servicio de Windows Application Driver la cual se minimizará.

ScreenPlay +BDD+WinAppDriver 2 - Configuración Inicial, Feature y StepDefinitons



Revisamos la ejecución en la parte Run y tendremos los métodos recomendados por cucumber para implementar en nuestra próxima clase StepDefinitions.



ScreenPlay +BDD+WinAppDriver 2 - Configuración Inicial, Feature y StepDefinitons

Crea una clase en el paquete **stepdefinitions** donde definiremos los métodos generados por cucumber, por ejemplo, “**FlightGuiStepDefinitions**”.

Clic derecho sobre el paquete **stepdefinitions** > **New** > **Java Class**.

Agregar los métodos propuestos a la clase de definiciones creada. Y nos debe quedar así después de pegar los métodos, importar las etiquetas de cucumber y eliminar las excepciones:

```
FlightGuiStepDefinitions.java x
1 package co.com.choucair.winappdriver.flightgui.stepdefinitions;
2
3 import cucumber.api.DataTable;
4 import cucumber.api.java.en.Given;
5 import cucumber.api.java.en.Then;
6 import cucumber.api.java.en.When;
7
8 public class FlightGuiStepDefinitions {
9
10     @Given("^that brandon wants to enter the flight gui desktop application$")
11     public void thatBrandonWantsToEnterTheFlightGuiDesktopApplication(DataTable arg1) {
12     }
13
14     @When("^he enters the data to buy the ticket$")
15     public void heEntersTheDataToBuyTheTicket(DataTable arg1) {
16     }
17
18     @Then("^he verifies the purchase with the message completed$")
19     public void heVerifiesThePurchaseWithTheMessageCompleted() {
20     }
21 }
```

Nos vemos en la próxima Guía...