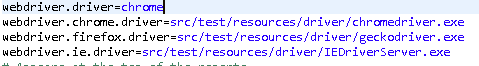
**¿Cómo construir un StepDefinitions más limpio?**

Normalmente nuestros ***StepDefinitions*** tienen una estructura como la siguiente:

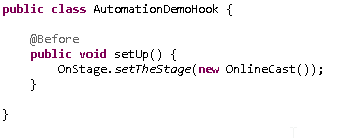


Haciendo uso del annotation @***Managed*** definimos un webdriver que posteriormente declararemos y usaremos. Sin embargo, nosotros ya habíamos definido el driver en nuestro archivo ***Serenity.properties***:

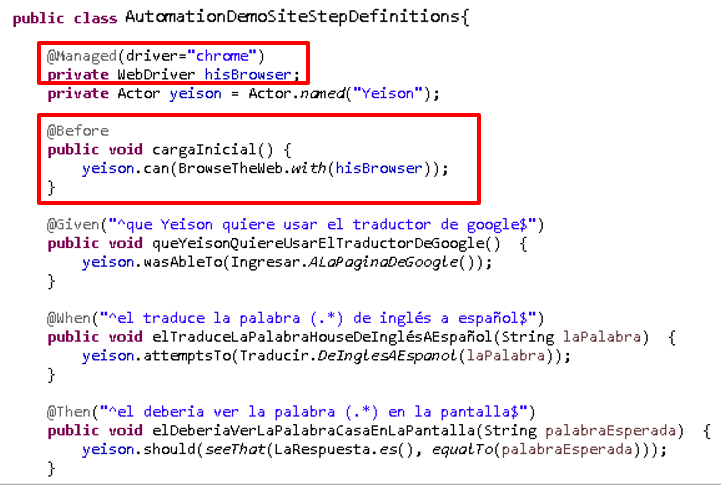


En el siguiente TIP veremos cómo tener un ***StepDefinition*** más limpio organizando nuestro código y haciendo uso del concepto “***Hook***” para nuestros Annotations @***Before*** @***Test***.

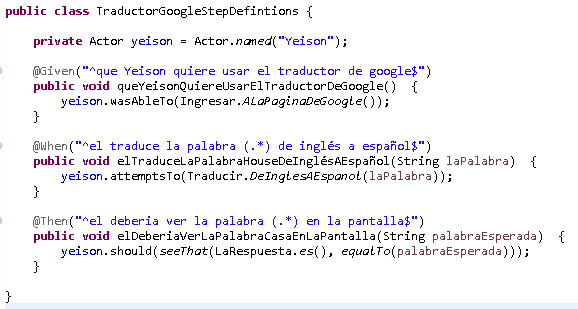
Un Hook será una clase que se ejecutará antes de iniciar nuestra prueba, es decir, hacemos un Hook para definir y ejecutar todas aquellas instrucciones que necesitemos para correr la prueba, por ejemplo, la definición de un driver y su asignación al objeto Actor. Para aplicar el concepto de Hooks, crearemos una clase en nuestro reto 1 Automation Demo, en el paquete Util. Esta clase podría llamarse AutomationDemoHook. (La palabra Hook no es obligatoria para el funcionamiento, solo es una buena práctica)



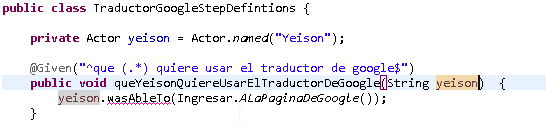
Le crearemos un método para hacer la configuración inicial del driver, en este caso hemos llamado al método setUp, cabe resaltar, que el nombre del método puede ser el que estimes conveniente. Usando la clase ***OnStage*** y su método ***setTheStage****,* con la instanciación de ***OnlineCast***, le diremos al Serenity que haga uso del driver que hemos definido en el ***serenity.properties***. (Para este ejemplo, el driver de Chrome). Esta porción de código se ejecutará automáticamente antes de ejecutar las instrucciones de nuestro @***Dado*** (@***Given***), quiere decir que ya podemos borrar de nuestro ***StepDefinition*** el siguiente bloque de código.



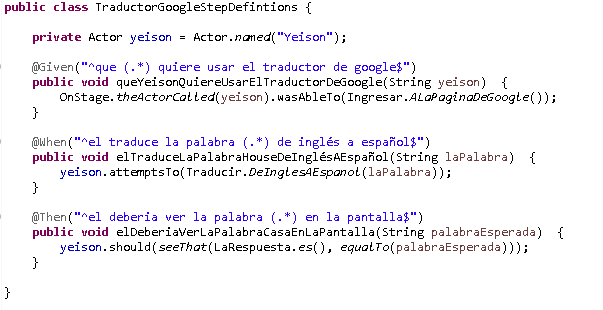
Teniendo como resultado un código más limpio.



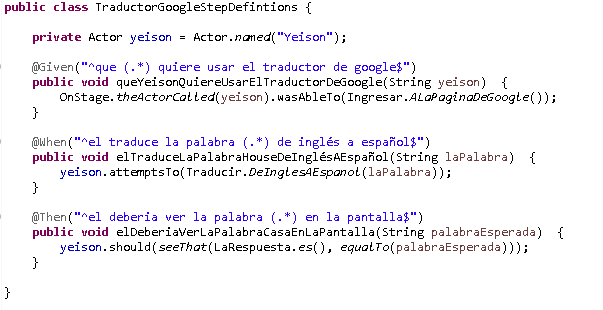
Sin embargo, aún tenemos la instanciación del Actor en nuestra clase ***StepDefinitions***, sería conveniente, hacer una expresión regular con el nombre del actor y usarlo en nuestros métodos @**Dado**, @**Cuando**, @**Entonces**. Para esto hacemos del nombre de nuestro actor una expresión regular.



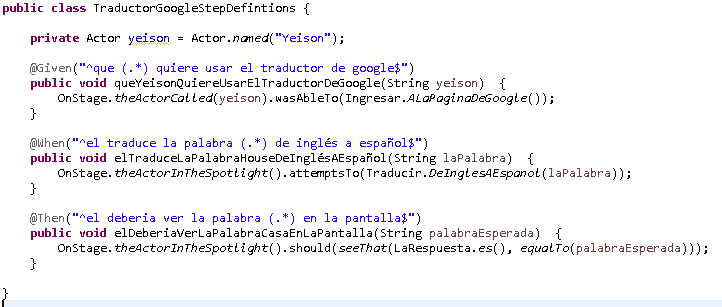
Al hacer esto vemos que nos marca un error, el cual corregiremos reescribiendo nuestra instrucción, haciendo uso de la clase ***OnStage*** y su método ***theActorCalled***.



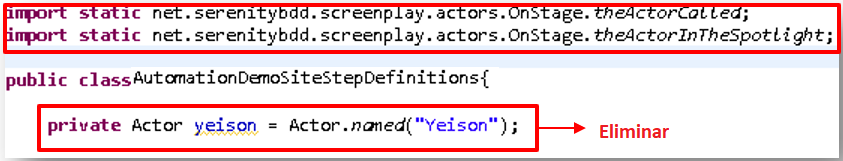
Ahora solo nos resta quitar las demás referencias al Actor yeison.



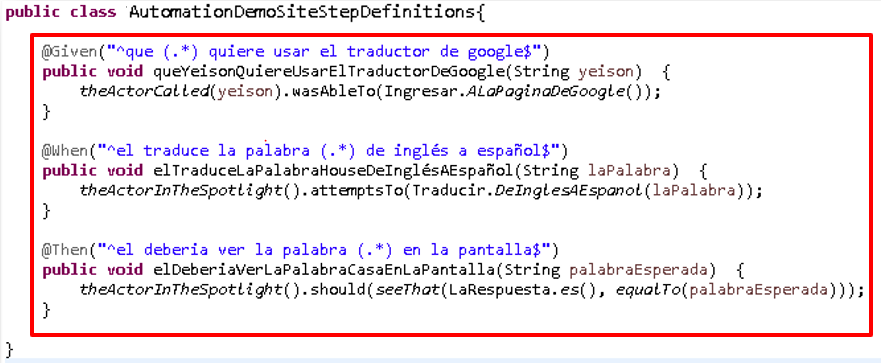
Para esto, usaremos nuevamente la clase **OnStage**, pero en esta ocasión usaremos el método “***theActorInTheSpotlLight***”. Para el @**Dado** usaremos siempre el método ***theActorCalled***, para el resto de nuestros pasos (@**Cuando** y @**Entonces**) usaremos el método ***theActorInTheSpotlLight****.*



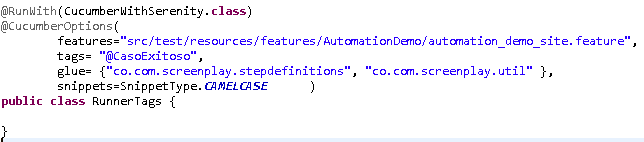
Ahora podremos borrar la línea inicial donde declaramos nuestro Actor e importar estáticamente la clase ***OnStage*** y los dos métodos que hemos usado.



Y tendríamos como resultado un StepDefinition que luce mucho mejor.



Por último, en nuestro **Runner**, le agregaremos a nuestra sentencia “**glue**” el paquete donde se encuentre nuestra clase ***AutomationDemoHook***



**Webgrafía**

<https://serenity-js.org/design/stage.html>

**¡Ahora inténtalo tú!**