PRÁCTICA 2

Pruebas automáticas e Integración Continua

Pablo López Parrilla
Doble Grado en Ingeniería Informática e Ingeniería de Computadores
10/06/2019
Ampliación de Ingeniería del Software
DNI:47313294v

Universidad
Rey Juan Carlos

Contenido

1-Introduccion y comentarios

Los conocimientos aprendidos y adquiridos con esta practica los considero de una gran importancia, pues el apartado de testing dentro de un proyecto no es menos importante que el propio desarrollo del mismo. En este caso hemos practicado la implementación de test unitarios, test con dobles y test corriendo el sistema con Selenium. Además, el utilizar una plataforma como Jenkins considero que nos acerca un poco mas al mundo laboral.

La práctica ha sido realizada sin problemas aparentes mas que los descritos abajo en la sección de comentarios, relacionados estos con Jenkins.

Para la ejecución de Jenkins se necesita al menos de un sistema con 4gb de RAM. La ejecución del mismo son aproximadamente 30 segundos, dependiendo de si ya se tienen descargadas las librerías a utilizar y del propio sistema.

2-Pruebas unitarias de la clase Board

Para la realización de la prueba de las dos funciones, he creado varias funciones en las que dibujo un tablero. Mas tarde en los test procedo a probar que las funciones nos dan el resultado que debiera.

En el caso de checDraw() buscamos darle un tablero en el que haya empate y otro en el que no. En el caso de getCellsIfWinner() lo que pretendemos es pasarle un tablero el cual sabemos cuáles serían los cellID ganadores y los comparamos con el la llamada a la función.

En ambos casos probamos tanto como que el jugador 1 gana y el jugador 2 también gana.

3-Pruebas con dobles de la clase TicTacToeGame

El funcionamiento del teste se compone en varias partes:

-Inicialización: Utilizaremos dos mock de la clase connection para hacer dobles del funcionamiento de la aplicación. Crearemos dos jugadores los cuales añadiremos al juego y comprobaremos de que estos son añadidos mediante la búsqueda de la invocación del evento JOINGAME. Esto se realiza en las dos conexiones para los dos jugadores.

-PruebaGanaJugador1: En este caso una vez ya inicializado el juego con los dos jugadores, se ira marcando cada casilla por cada jugador, y comprobando que dicho marcaje ha sido notificado

en ambas conexiones. Finalmente se buscará la que se haya notificado el evento GAMEOVER. Además, comprobamos que las celdas ganadoras corresponden a las que nosotros habíamos introducido. También comprobamos el id del jugador ganador.

- PruebaGanaJugador2: lo mismo que el anterior, pero ganando el segundo que comienza

-PruebaEmpate: en este caso forzamos un empate entre los dos jugadores. Comprobaremos también que se haya notificado el GAMEOVER, además en caso de que no haya ningún ganador significara que el argumento evento no esta inicializado y por lo tanto es NULL.

Para la realización de los test nos ayudamos de argThat y hasitems para verificar los eventos, así como de reset para limpiar las conexiones y finalmente argumentCaptor para el caso en el que sean varios los parámetros (un objeto clase Object).

4-Pruebas de sistema de la aplicación

La prueba del sistema ha sido la que mas recursos de tiempo ha necesitado. Usando como plantilla lo aprendido en clase, he seguido este esquema:

-BeforeClass: creamos el webdrivermanager y arrancamos la aplicación.

-Before: inicializamos los drivers y las esperas de los mismos.

-After: cerramos los drivers

-AfterClass : apagamos la aplicación.

-Login: se trata de una función que pasándole un String y un driver, nos crea en la aplicación un jugador con ese nombre.

-PruebaLogin: prueba que los dos jugadores se hayan creado correctamente utilizando la interfaz grafica en la cual podemos encontrar los nombres en el apartado “pXScore”

-PruebaGanadorJugador1: testearemos que en el caso de que el jugador1 gane, para ello clicamos en cada uno de los turnos la casilla correspondiente, esperando a que el jugador contrario observe que ya se ha clickado para que no intente clickar antes de que se le notifique. Una vez terminamos, esperamos la alerta y comprobamos que en esta dice que gana el jug 1.

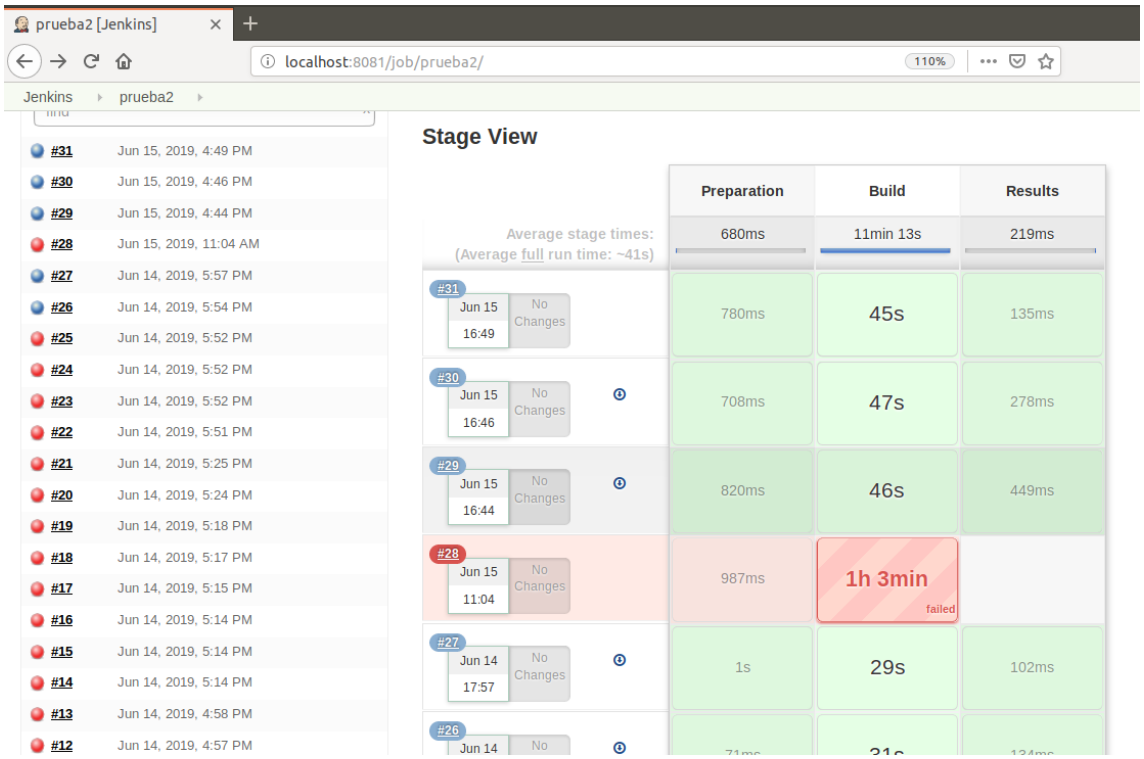
-PruebaGanador2: lo mismo que PruebaGanador1, pero con el jugador 2

-PruebaEmpate: provocamos un empate y comprobamos que la alerta que el sistema provoca corresponde a “Draw!”

5-Integracion Continua

Para la configuración de Jenkins hemos usado la guía que se incluyen en las transparencias, añadiendo pequeños cambios en el configure. En este caso para hacer funcionar un código desde un repositorio local lo que hago es copiar dicho repositorio al workspace de la tarea y extraer el .zip, ya Jenkins se encarga de ejecutarla correctamente.

Finalmente se ha sacado el log de la consola y se comprueba que funciona la ejecución de los test con “mvn test”



Comentarios

En la realización de esta practica me he encontrado como uno de los mayores retos el enfrentarme a las dependencias en java. En ninguna otra asignatura había tenido que modificar el pom.xml con anterioridad, por lo menos no en la misma cantidad que en esta.

Cuando por fin he conseguido que todos los test funcionen, adecuando las versiones de las librerías al código utilizado para las mismas, me he enfrentado a Jenkins. El problema que yo tenia es que en el enunciado se pide que Jenkins tiene que funcionar en un sistema Linux, en mi caso había realizado toda la practica en un windows10.

Procedo a instalar un ubuntu18.10 desde cero, la instalación del jdk8 ha sido, aunque parezca mentira lo que mas tiempo me ha llevado. Finalmente, después de varios videotutoriales lo he conseguido.

Configuramos Jenkins como se pide en el enunciado y procedemos a ejecutar los test de ejemplo de la teoría dada en clase. Múltiples errores, ningún código funciona y la configuración de java no era la correcta. Una vez conseguido que funcionen los ejemplos pasamos con la practica en si.

La practica parece ser que se queda pillada y da algún error de tipo Chrome driver failed…. En este momento procedo a intentar realizar las mismas funciones desde el Windows, después de configurarlo todo parece que funciona. Pues el error en mi caso por el cual no llegaba a completarse la tarea era por motivos de Memoria Principal. El sistema sobre el que estaba ejecutando Jenkins tenía 3Gb de RAMjenkins.t, insuficiente en el momento de abrir los web drivers de Chrome. Ha sido necesario ejecutar dicho programa en un sistema de 4gb, descubriendo que la tarea en ejecución en Ubuntu conseguía un consumo total de 3,3GB de RAM