# Ejercicio Repaso PPSS

## S07 – P07

1. En este caso podemos realizar las pruebas de integración, porque estaríamos probando la interacción entre distintas unidades del sistema, además de que la base de datos se suele encontrar en una capa distinta a la del código a probar.
2. Utilizaríamos el framework de DBUnit, que es una extensión de Junit enfocado para las bases de datos. Al pom se añadirían las dependencias de:
   * 1. Org.dbunit
     2. Mysql

También se tendría que añadir el plugin de Failsafe para la ejecución de los test de integración.

1. El código de este ejercicio se encuentra en el fichero *S07-P07/ITTestDBUnit.java.*

## S08 – P08

1. **Durante la práctica hemos usado el diseño de casos de prueba basado en escenarios, ya que se nos iba describiendo paso a paso como teníamos que ir usando la aplicación web para testarla.**
2. **No, porque estas pruebas están basadas en un modelo funcional de la aplicación y no es el código lo que se prueba, si no el correcto funcionamiento de la aplicación construida y en fase de producción. Además, al no ser código java y no indicar ninguna dependencia en el pom del proyecto, esto daría paso a un error durante la construcción del proyecto.**
3. **El usuario accederá a la página de Bitbucket.org, una vez aquí el usuario se tendrá que loguear con sus credenciales. Una vez que el usuario se haya logueado se mostrara su pantalla de inicio y a través de esta accederá a todos sus repositorios, y seleccionara el repositorio de la asignatura de ppss y comprobara la existencia de la carpeta P08 y que esta no se encuentra vacía y con los ficheros correspondientes.**

## S09 – P09

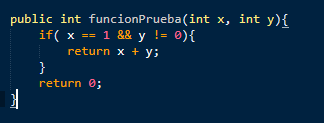
1. Si que se podrían implementar, pero de forma distinta ya que ambos testean el funcionamiento de páginas Web y WebDriver de manera mucho más sencilla, eficaz y con menos limitaciones que Katalon Recorder.
2. Si, ya que estas pruebas tenemos que incluir sus dependencias en el pom del proyecto, por lo tanto, a la hora de construir el proyecto, este lo tendrá en cuenta y no dará error.

Los ficheros de código correspondientes al ejercicio de implementación de un caso de prueba, se encuentran en la carpeta S09-P09.

## S10 – P10

1. Para que, a través del navegador, podamos realizar las pruebas y que así Meter pueda grabar las peticiones HTTP generadas tal y como se envían al servidor.
2. En la práctica 10 creamos un servidor proxy y conectamos el navegador Firefox a este servidor para poder recoger toda la información sobre las peticiones que realizamos a la aplicación web JPetStore.
3. Desde el menú contextual del banco de trabajo, seleccionamos “Añadir -> ElementosNoDePrueba -> Servidor Proxy HTTP”, durante la configuración de este, hay que asegurarse de que el puerto que vamos a usar para este servidor no está ocupado. Una vez creado hay que configurar el navegador que vayamos a utilizar para realizar las pruebas, para la configuración de este, tendremos que ir a sus opciones y poner que la conexión la realice a través de un servidor proxy e indicar los datos del servidor proxy que hemos creado anteriormente. Una vez hemos configurado el servidor y el navegador desde la aplicación de JMeter le daremos al botón de Arrancar, para poder navegar desde el servidor que hemos creado y poder recoger todas las peticiones que realicemos sobre la página web que estamos testando

## S11 – P11

1. Con los siguientes casos de prueba conseguimos en el código aportado en la imagen una cobertura de condiciones 100% pero no de líneas ya que el *return x+y;* nunca llegaría a ejecutarse.
   1. C1: (X = 1; Y = 0)
   2. C2: (X = 2; Y = 3)
2. Las tres métricas que aparecen en los informes de Cobertura son:
   1. *Line Coverage:* Indica el número de líneas de código que se ejecutan en los test.
   2. *Branch Coverage:*Comprueba si se ejecutan todas las condiciones del código a probar en cada una de sus vertientes. Y el resultado es el porcentaje de condiciones cubiertas frente al total de condiciones.
   3. *Complejidad ciclomática:*Calcula el número de casos de prueba necesarios para cubrir todos los caminos linealmente independientes del grafo de control de flujo del código a probar.

## S12 – P12