Laboratorio 0: Uso del Debugger en Eclipse

Introducción

Cuando hay errores en las aplicaciones hay que depurarlos o eliminarlos, para lo cual se requiere detectar dónde y por qué ocurren, antes de corregir la implementación que evite que ocurran de nuevo. Existen herramientas imprescindibles para los desarrolladores, que ayudan en la tarea de detección de errores en el software: son los programas depuradores o debuggers. Un programa depurador sirve para ejecutar paso a paso el código fuente con el fin de buscar el punto exacto donde se ha cometido un error de programación (también conocido como bug).

En este laboratorio utilizaremos el código inicial del proyecto Bets. No vamos a intentar arreglar un error en concreto, sino que veremos cómo utilizar el debugger: definición de puntos de parada, ejecución paso a paso y visualización de los valores de las variables.

Objetivos

El objetivo de este laboratorio es saber utilizar el debugger incluido en la herramienta de desarrollo Eclipse.

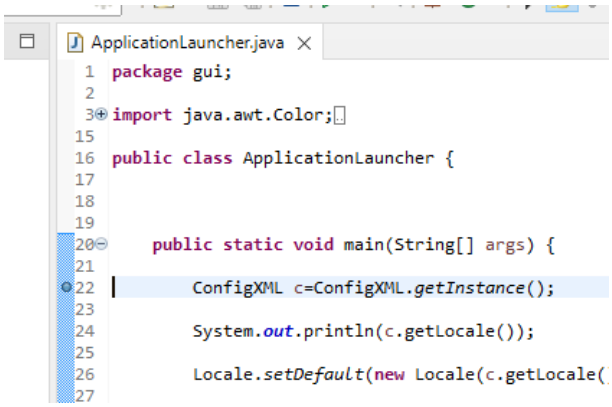
Pasos a seguir para utilizar el debugger

1. Definición de puntos de parada (breakpoints)

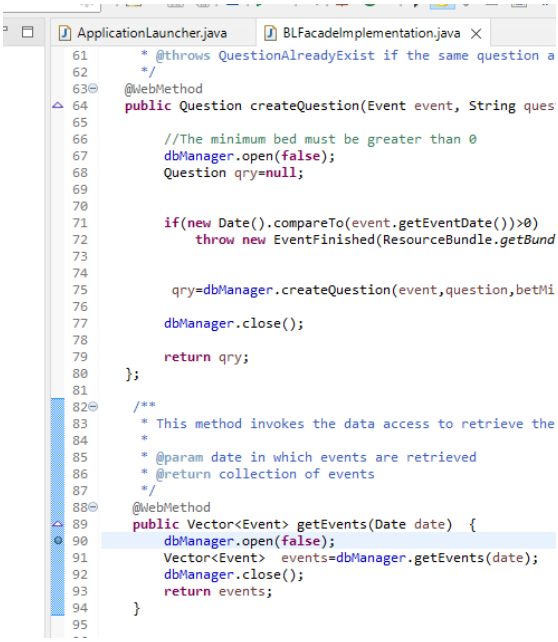
En el programa se pueden definir unos puntos de parada, para que, a partir de ellos, se pueda realizar esa ejecución paso a paso. Para definir un punto de parada (breakpoint) en el código:

Posicionarse en una línea concreta del código fuente (sobre el número de línea)
=> Hacer click derecho => Toggle breakpoint
[=> O si no, tras posicionarse en la línea => Hacer doble Click izquierdo]

Definir el siguiente breakpoint en la línea 22 de la clase gui.ApplicationLauncher:



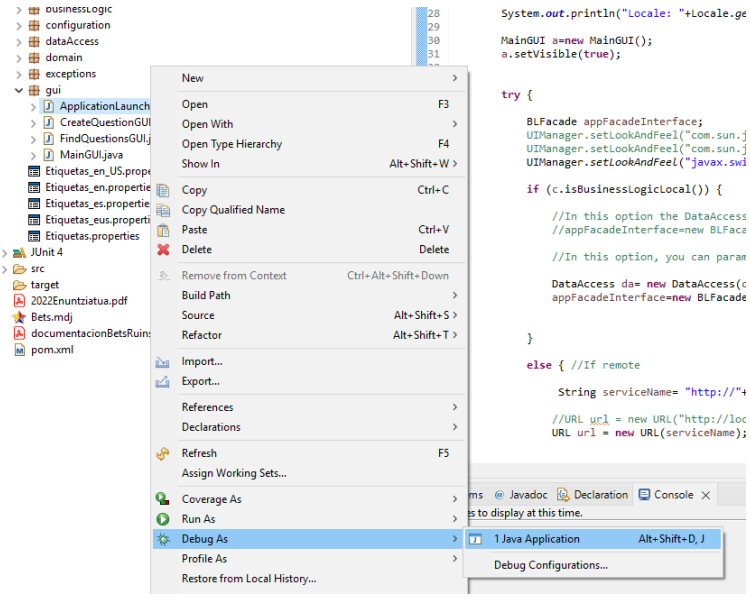
Y otro breakpoint en la línea 90 de la clase businessLogic.BLFacadeImplementation:



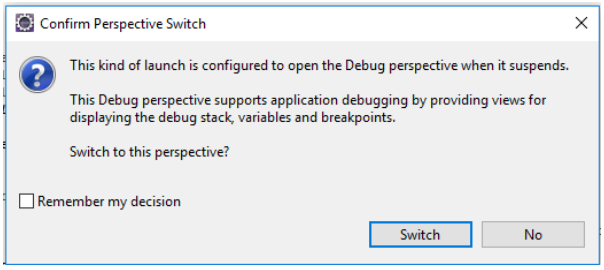
2. Ejecutar la aplicación en modo debugger

Ejecutaremos la aplicación en modo debugger, la cual se ejecutará normalmente hasta que se alcance alguno de los breakpoints definidos. Como la clase principal es ApplicationLauncher, esa es la que se ejecutará en modo debugger:

Seleccionar la clase ApplicationLauncher => Click dcho. => Debug As => Java Application

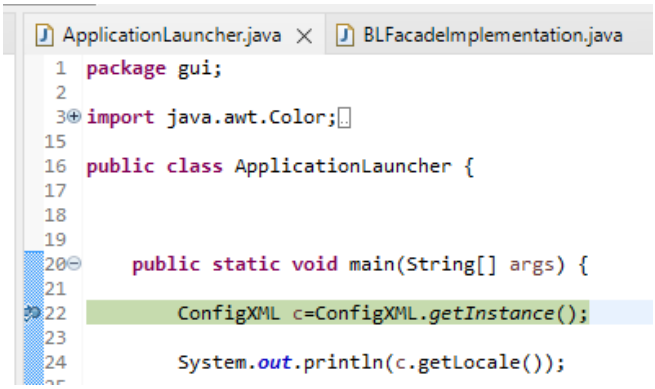


Cuando vaya a ejecutarse la instrucción de un breakpoint entonces Eclipse solicitará cambiar la perspectiva actual a la de Debug => Pulsar Switch



3. Ejecutar las instrucciones del código fuente paso a paso

En este punto, se puede ver en la perspectiva de Debug que la ejecución se ha detenido en el primer breakpoint encontrado:

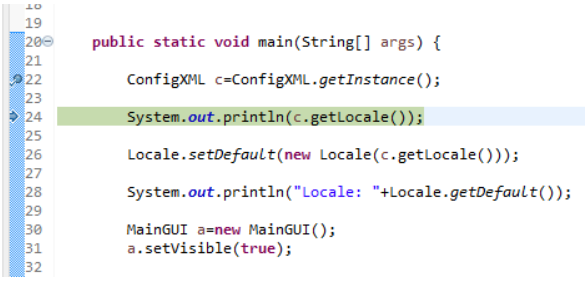


Y también se puede comenzar a controlar la ejecución paso a paso con las siguientes opciones:

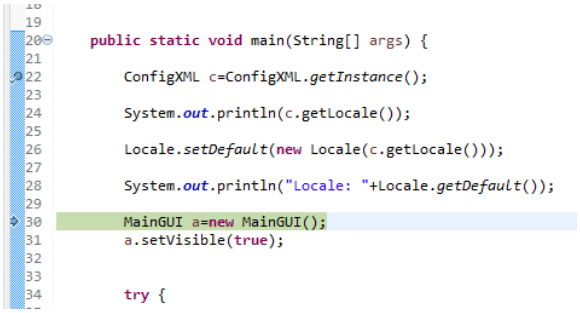
Haciendo click en la opción “Step Over” o pulsando F6, indicaremos que se ejecute la siguiente instrucción de una vez, esto es, sin entrar dentro de la llamada al método *getInstance*()

```
ConfigXML c=ConfigXML.getInstance();
```

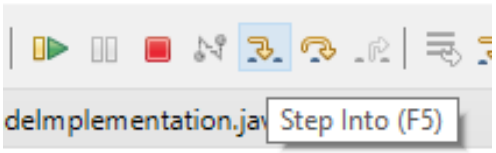
A continuación puede verse que el debugger se ha vuelto a detener en la siguiente instrucción: `System.out.println(c.getLocale());`



Después de pulsar 3 veces la opción “Step Over (F6)” el debugger se detendrá en la siguiente instrucción: `MainGUI a=new MainGUI();`

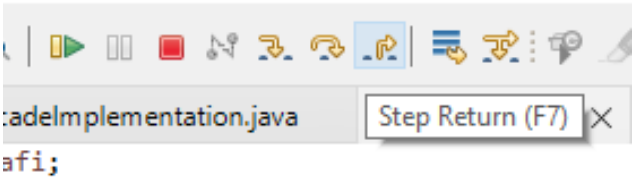


Haciendo click ahora en la opción “Step Into” o pulsando F5



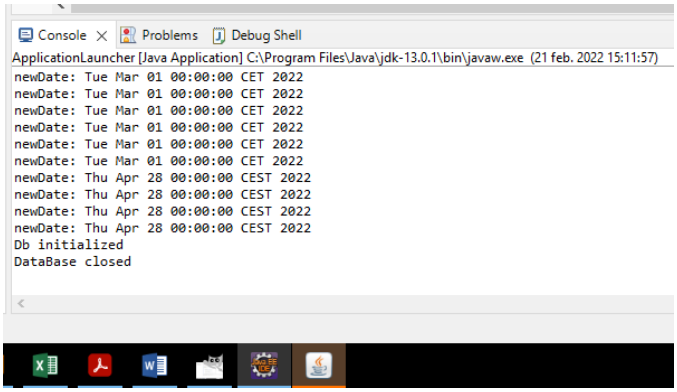
se indica al debugger que se ejecute la siguiente instrucción entrando dentro de la llamada al método constructor `MainGUI()`, tal y como se muestra en la siguiente figura:

Podríamos continuar con el debugger dentro de este método, o podríamos regresar al punto desde donde se llamó al método para continuar desde ese punto anterior. Para ello tenemos la opción “Step Return (F7)”:

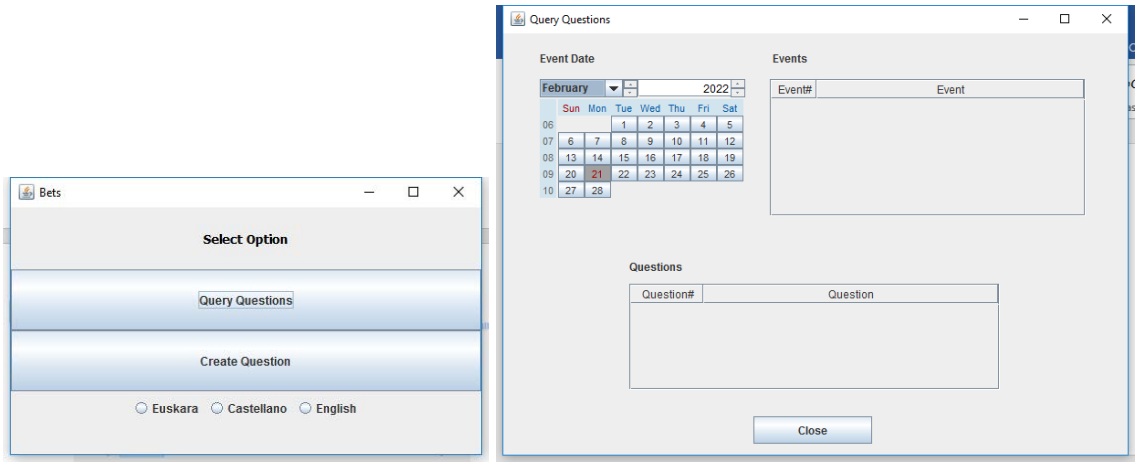


También podríamos dejar que la aplicación continúe su ejecución hasta el siguiente breakpoint. Eso se hace con la opción “Resume (F8)”

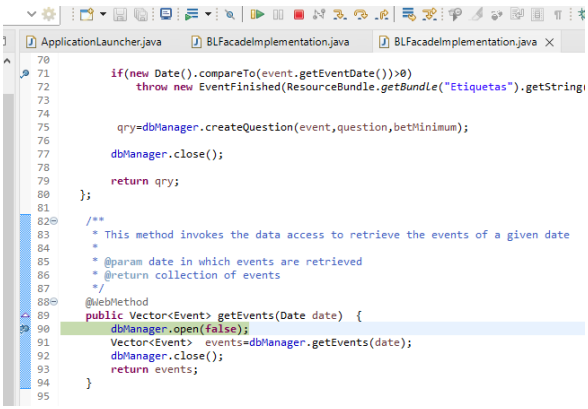
Tras ejecutar “Resume (F8)”, veremos que la aplicación sigue su ejecución, y ha creado la ventana principal, que se debe activar (pulsando el icono de Java en la barra de herramientas):



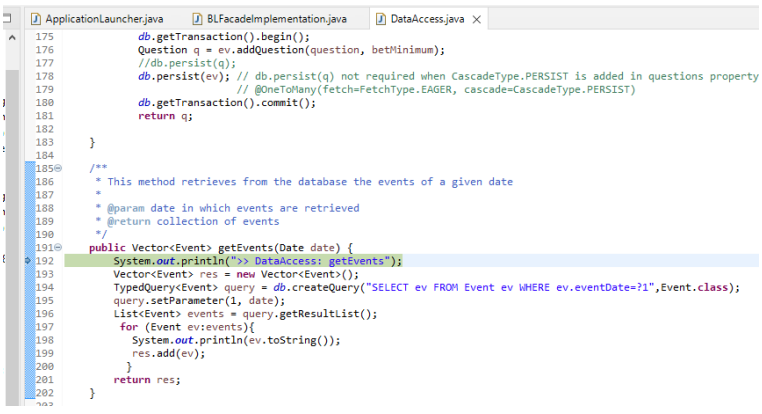
Para llegar al siguiente breakpoint (definido en el método getEvents de la lógica del negocio) deberemos provocar su ejecución.
Para ello: pulsar botón “Query Questions” => Seleccionar fecha en el calendario



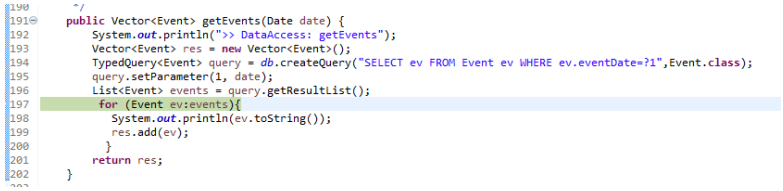
Ahora el debugger muestra la siguiente instrucción a ejecutar:`dbManager.open(false);`



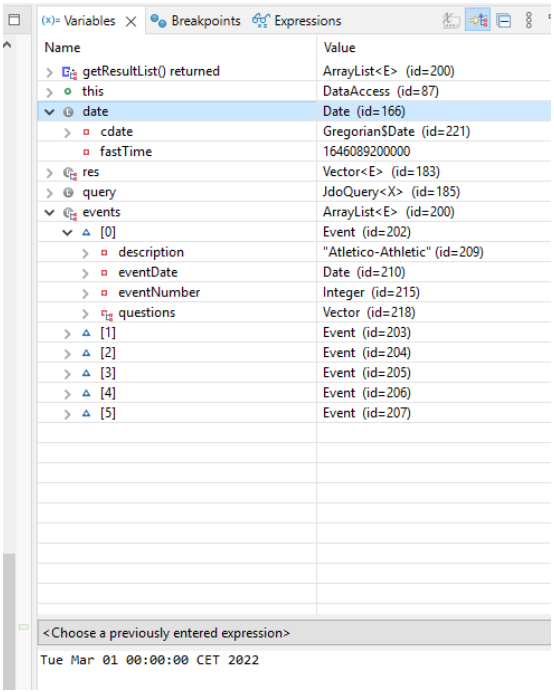
Pulsando F6 => F5, se entrará dentro de la llamada al método getEvents(date) de DataAccess:



Si se pulsa 6 veces F6 hasta llegar a la instrucción `query.getResultList();`



En este punto puede verse el contenido de las variables (events, query, date,…), que podría ser el siguiente, si se hubiera seleccionado una fecha con eventos (1/03/2022, en este caso).



Si no aparece la pestaña con las variables: Window =>Show View => Variables

Si no hay eventos en la fecha actual, se puede hacer “Resume (F8)” en el debugger y escoger una fecha con eventos en el calendario.

Por último, para terminar la ejecución del debugger está la opción “Terminate (Ctrl + F2)”