

Manual de Depuración en Eclipse.



Contenido

1. Utilidades para depurar.	1
2. Comandos del depurador:.....	2
3. Vistas útiles dentro de la perspectiva de depuración:	3









1. Utilidades para depurar.

Estas herramientas nos permitirán depurar nuestro programa en Eclipse de manera eficiente.

- **Perspectiva de depurador:** Nos permite utilizar las herramientas de depuración.
 - Ruta: Window -> Perspective -> Open Perspective -> Debug.
- **Mostrar número de las líneas de código:** Podremos tener un indicador intuitivo de dónde se encuentran los errores encontrados por el depurador.
 - Ruta: Window -> Preferences -> General -> Editors -> Text Editors -> Casilla "Show Line Numbers".
- **Filtrar métodos en la depuración:** Nos permite filtrar las clases que no nos interese inspeccionar a la hora de depurar un programa, por ejemplo las clases de Java.
 - Ruta: Click derecho encima de la clase en la vista "Debug" -> Edit Step Filters -> Casilla "Use Step Filters" -> Casilla "Java.*" o la que nosotros deseemos.
- **Breakpoint:** Señal utilizada para indicar al depurador dónde debe parar la ejecución del programa.
 - Ruta: Doble click sobre la barra lateral izquierda del editor de código.


2. Comandos del depurador.

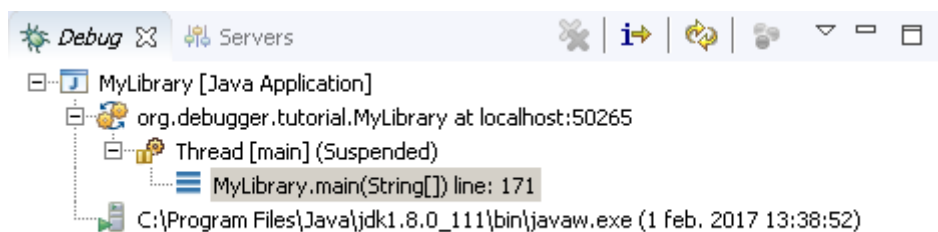
Los comandos del depurador nos permiten manejar nuestra herramienta de depuración. Muchos de ellos podemos encontrarlos en la pestaña “Run”, en las barras de herramientas o usarlos a través de atajos.

Comando	Función	Atajo
Debug	Nos permite iniciar el proceso de depuración.	 / F11
Resume	Permite ejecutar todo el código hasta que el depurador encuentre un breakpoint o se acabe el programa.	 / F8
Terminate	Obliga a terminar la depuración.	 / Ctrl + F2
Step Into	El depurador se introduce en el método seleccionado por el programa.	 / F5
Step Into Selection	El depurador se introduce en el método seleccionado con el ratón.	Ctrl+alt+click
Step Over	Ejecuta la línea de código actual y se para en la siguiente.	 / F6
Step Return	Termina el método actual y vuelve al método que lo llamó.	 / F7
Use Step Filters	Nos permite activar o desactivar los filtros de depuración.	 / Shift + F5
Run to Line	Nos permite ir rápidamente hacia la línea seleccionada por el cursor sin la necesidad de utilizar un breakpoint.	Ctrl + R
Inspect	Nos permite saber los valores de una variable o una expresión en un <u>momento determinado</u> . Selecciona el código a inspeccionar y pulsa el atajo. Pulsa de nuevo el atajo para añadirlo a la vista “Expressions”. Muy útil en la vista “Display”.	Ctrl+Shift+I
Drop to Frame	Se utiliza para volver atrás en el código, teniendo en cuenta que las variables mantendrán sus valores.	Selección de la línea + 

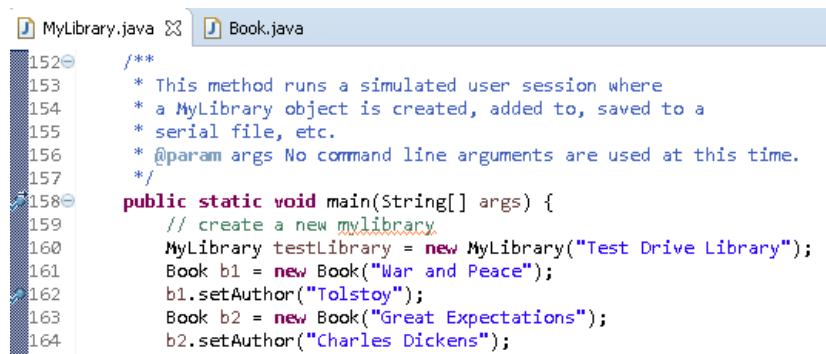
3. Vistas útiles dentro de la perspectiva de depuración.


Para trabajar en el depurador de eclipse, utilizaremos distintas vistas donde podremos encontrar las herramientas necesarias. Las más destacables son:

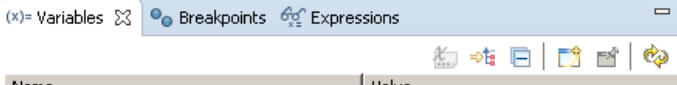
- **Debug:** En ella podremos visualizar los métodos llamados en nuestra depuración. Cada uno de los iconos  representa un método abierto. Si pinchamos en uno diferente en el que estamos, el resto de perspectivas cambiarán para ofrecernos información acerca de dicho método.



- **Editor de código:** En esta vista podremos observar nuestro código e incluso modificarlo durante la depuración. Si haces esto último, ¡Recuerda guardar la modificación si quieres que se aplique cuando reanudes la depuración!

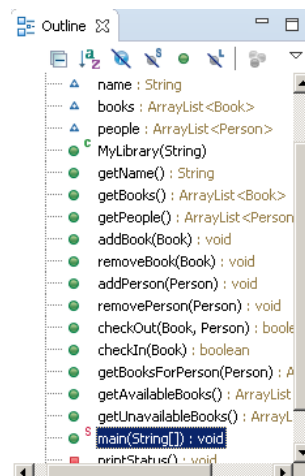


- Variables: Podremos observar las variables y los valores de éstas en el método actual. Si la variable tiene más de un atributo, podemos verlos desplegando la lista con el botón . Además también existe la posibilidad de cambiar el valor de las variables con click derecho -> “Change Value” por si necesitamos ver el funcionamiento del programa con otro valor.

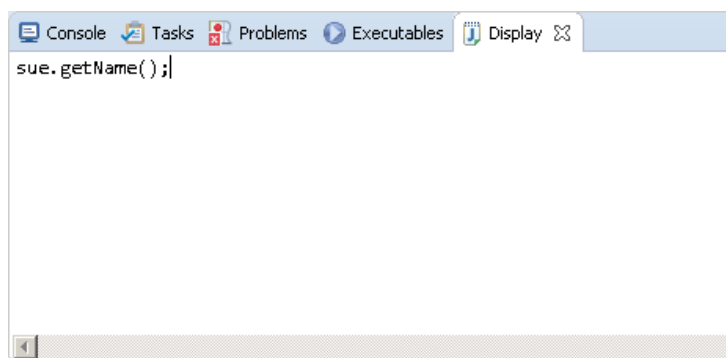




Name	Value
args	String[0] (id=17)
testLibrary	MyLibrary (id=20)
b1	Book (id=25)
author	"Tolstoy" (id=70)
person	null
title	"War and Peace" (id=26)

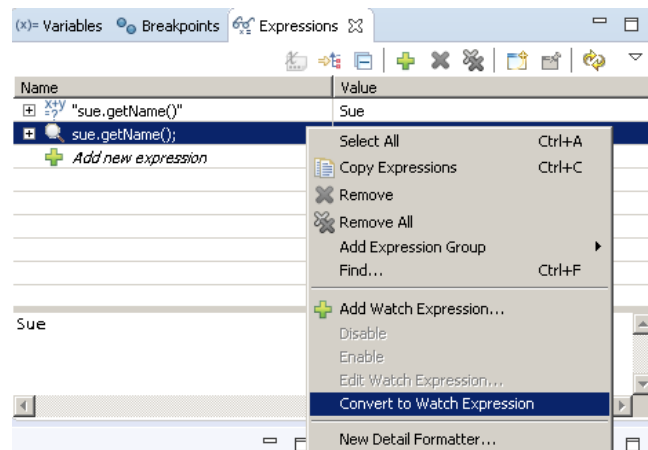
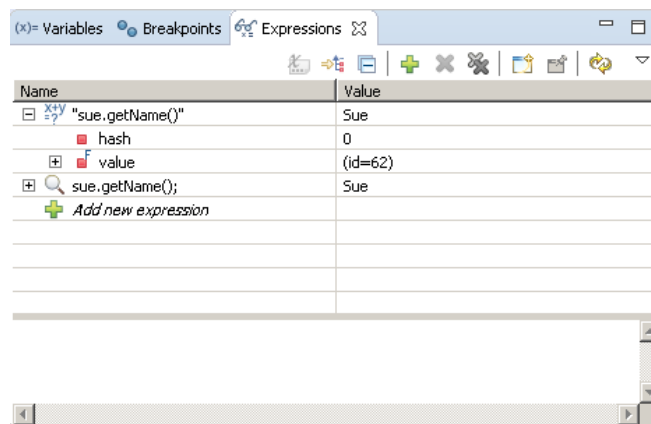
- Outline: Nos sirve para visualizar todos los métodos que contiene nuestro fichero.



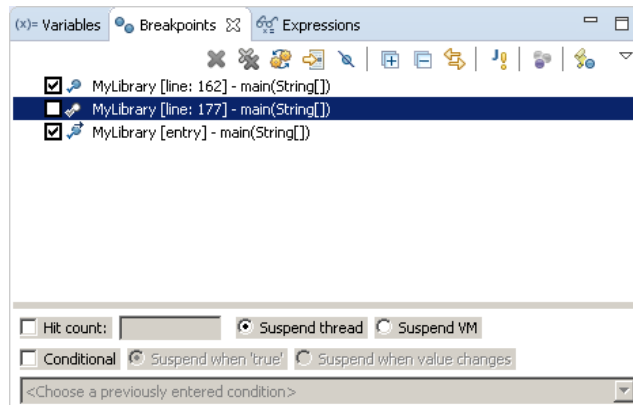
- Display: En esta vista podremos introducir líneas de código, muy útil para cuando queremos conocer el valor de variable (Ejemplo: Sefran.getAge()). Si dicho código se encuentra en la vista de edición, podemos seleccionarlo con el ratón -> click derecho -> display, para inspeccionarlo más rápidamente.



- Expressions: Esta vista nos sirve tras ejecutar expresiones de código con la vista o el atajo de “Display” y nos permite conocer su resultado. Podemos hacer que estas expresiones sean:
 - Dinámicas: Su valor cambia durante el recorrido de la depuración. Se representan con el icono  y se le denominan “Watch Expressions”
 - Estáticas: No cambian su valor durante la depuración. Se representan con el icono  y se conocen como “Inspecciones”. Podemos añadir inspecciones utilizando el comando “Inspect” y convertirlas a watch expressions más tarde con click derecho -> “Convert to Watch Expression”.

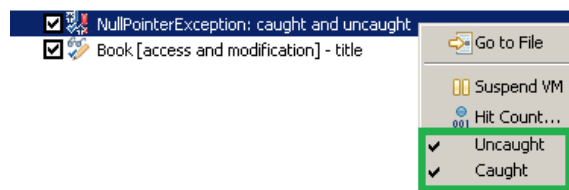


- Breakpoints: Nos permite visualizar los breakpoints colocados en el editor de código. Podemos marcar y desmarcar aquellos breakpoints que nos convengan.




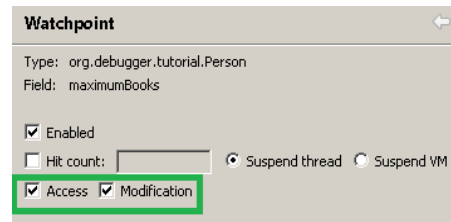
Estos breakpoints pueden tener distintos condicionantes, por ejemplo:


1. Podemos decir que programa pare cuando intercepte excepciones de java en el botón . Tras pulsar el botón sólo necesitaremos indicar qué tipo de excepción queremos que se intercepte. Una vez creada, podremos seleccionar con el click derecho si quieres que se pare en las excepciones capturadas y/o no capturadas. Lo habitual es seleccionar sólo las no capturadas, ya que las capturadas están controladas por el programador.



2. Si la posición de nuestro breakpoint está dentro de un bucle, podemos indicar el número de veces que queremos que salte, útil cuando sabemos que dicho bucle tiene un problema en un valor en concreto. Se accede con click derecho sobre el breakpoint -> Breakpoint Properties -> Hit Count -> indicar número de veces.
3. Si queremos que nuestro breakpoint se pare cuando ocurra un evento en concreto dentro de una variable, podemos indicarlo en click derecho sobre el breakpoint -> Breakpoint Properties -> Enable Condition -> escribimos la condición a cumplir. (Ejemplo: `this.getValorX() = 0`)

4. Si queremos que el depurador pare cuando cambie el valor de una variable, usaremos un breakpoint especial denominado “Watchpoint”, que se identifican con el icono . Para crearlos debemos irnos a la declaración de la variable en su clase correspondiente (Ejemplo: Atributo “private String Name” de la clase “persona”) y marcar como si fuese un breakpoint. Si observamos las propiedades de los watchpoints (click derecho encima del watchpoint en la vista “breakpoints” -> breakpoint properties) nos permiten pararnos en las variables cuando se acceda a ellas o sólo cuando se modifiquen.



5. Por último tenemos el “Class Prepare Breakpoint”. Simplemente nos notifica de la primera vez que una clase se ejecuta. Se crea colocando un breakpoint en la línea de la declaración de la clase. Se simboliza con .