



- 1 Objetivos
- 2 Taller
- 3 Debug
  - 3.1 Ejemplo 1
  - 3.2 Ejemplo 2
- 4 CVS
- 5 Trabajo a hacer
- 6 Cosas a tener en cuenta

---

## OBJETIVOS

- ▶ Perder el miedo al entorno
- ▶ Tocar un poco las principales cosas que hay para que después quien quiera se lo mire con calma.
- ▶ No entraremos a configurarlas, pero todo es configurable.
- ▶ Muchas cosas son útiles una o dos veces al día, no más. Ahora las veremos, aunque no haremos un uso extensivo.
- ▶ Es una introducción guiada, rápida y por encima de las diferentes posibilidades que ofrece Eclipse. No se ven todas, y muchas sólo se ven por encima. Básicamente es para perder el miedo al entorno y ver lo sencillo que resulta.

- 1 Disponer de un JDK instalado. Si solo tenemos un JRE no podremos acceder a la documentación de la API de java.
- 2 Descomprimir Eclipse
  - 2.1 tar -xzf eclipseblabla
- 3 Abrir el navegador del sistema, abrir la carpeta donde lo hemos descomprimido, y clicar dos veces sobre 'eclipse'.
- 4 set default workspace and don't ask again.
- 5 Pasar de los asistentes que muestra y clicar Workbench (se pueden recuperar desde la ventana de ayuda)
- 6 Maximizar la ventana, porque sino la barra de debajo probablemente no se vea
- 7 Jugar con las ventanas, perspectivas y vistas (se pueden desplazar, apilar, ocultar, window -> show view)
- 8 Creamos el proyecto HolaMundo
  - 8.1 "New Java Project" HolaMundo
  - 8.2 Allow output folders
  - 8.3 Marcar "Create separate source and output folder"
- 9 Desplegamos el árbol de la izquierda clickando sobre HolaMundo dentro de src, botón derecho, new Package 'jornadesPI'. Dentro del package creado hacer botón derecho New Class HolaMundo (observar el aviso que sale en la parte superior si escribimos un nombre incorrecto)
  - 9.1 Marcar checkbox "public static void main"
- 10 Observar marcas TODO que aparecen en la lista de tareas.
- 11 Si la ventana de tareas no está abierta, ShowView->Other->Basic->Task view
- 12 Observar las marcas de Folding. Muestran y esconden partes de código.
  - 12.1 Atención a las búsquedas.
- 13 Poner el código correspondiente para escribir una frase en la consola.
- 14 Escribir System. Observar como nos despliega las posibilidades
- 15 Si no nos lo despliega, podemos activar el content assistant (CTRL + espacio)
- 16 Ver 'content assistant' para parámetros (CTRL+SHIFT+espacio)
- 17 Borrar algún carácter de la instrucción para probar QuickFix's varios (nombres mal escritos...) (CTRL+1)
- 18 Ver los errores de compilación en la vista de compilación
  - Windows->ShowView->Problems
- 19 Encontrar la pareja del corchete (doble click)
- 20 Para ejecutar
  - 20.1 Clicamos el botón Play
  - 20.2 Seleccionamos 'Java Application'
  - 20.3 new
  - 20.4 Si la clase que estamos editando en ese momento contiene un método main nos la pondrá directamente. Si no la podemos buscar.
- 21 A continuación añadimos comportamiento para leer de la consola
- 22 Escribimos (utilizando CTRL+Espai)
 

```
BufferedReader in;
in = new BufferedReader (new InputStreamReader (System.in));
```

  - 22.1 i así podremos hacer una prueba leyendo unos caracteres `in.readLine()`;
  - 22.2 con CTRL+1 (para activar Quick Fix) se puede definir la variable local y envolver con el try/catch correspondiente
- 23 y escribimos en la consola el resultado

**Opciones de usabilidad (van a gusto del consumidor)**

- ▶ Activar SMART Typing
- ▶ Mark occurrences (rotulador amarillo)
- ▶ Activar la navegación sincronizada del panel. Ver que cuando cambio la clase editada esta queda seleccionada en el árbol.

### 3.1.- Ejemplo 1

- 1 Hacemos doble click al lado de la primera línea de nuestro código.
- 2 Clicamos en el escarabajo (bug)
  - 2.1 Nos avisa que toca cambiar la perspectiva, le decimos que sí y que lo haga siempre
- 3 Empieza la ejecución
- 4 Podemos ver los hilos, avanzar un paso, entrar a la función, avanzar hasta el cursor... y también Drop to frame (si la MV lo permite). Atención que los valores no se actualizan
- 5 Breakpoints condicionales, con contador, para escritura o para cambio, por valor

### 3.2.- Ejemplo 2

1. Anulamos todos los breakpoints
2. Hacemos doble clic al lado del system.out (para que aparezca un punto rojo, que es el breakpoint)
3. Clicamos el escarabajo
4. Le decimos que sí, que nos abra la perspectiva y que se acuerde.
5. Escribimos alguna cosa en la consola (es lo que nos pide el programa)
6. Cuando se ha parado, en la vista de variable seleccionamos con el botón derecho la del string y le decimos 'change value', cambiamos el botón y hacemos 'resume'
7. El resultado que nos muestra es el nuevo valor introducido con el programa parado y no el que se ha introducido realmente durante la ejecución

- 1 Primero jugar con la opción "Compare local history" y similares. Por ejemplo para recuperar el código que habíamos puesto antes de añadir el bucle.
- 2 Y una vez descubiertas las ventanas de comparación probar con el sistema CVS
  - 2.1 Window-> open Perspective-CVS
  - 2.2 Botón derecho sobre el espacio de la izquierda
  - 2.3 New repository location
  - 2.4 URL ???ATENCIÓN FIREWALLS
  - 2.5 Location /usr/local/cvsroot IMPORTANTE, no hay / al final
  - 2.6 user botón i password botón
  - 2.7 Desde la perspectiva Java importar el proyecto
  - 2.8 ATENCIÓN: Son dos pasos, "check out" del CVS i después crear un proyecto Java con el código que hemos sacado (y dejarlo relacionado con el del CVS). Por tanto parecerá que cuando acabamos el asistente volvemos a empezar
  - 2.9 Marcar "use an existing module" y seleccionar "botonera"
  - 2.10 Marcar Finish
  - 2.11 Entonces se abre el asistente de New Java Project, le ponemos el nombre que queramos (botonera) y continuamos
- 3 Ya tenemos el código
- 4 Hacer un cambio en el botón correspondiente
  - 4.1 Vemos que en el árbol de exploración nos ha añadido una marca, el símbolo > que indica que el fichero ha cambiado
  - 4.2 Entonces hacer botón derecho team/sincronize para enviar el cambio al CVS
  - 4.3 Desplegar el árbol de la izquierda, y doble click para ver los cambios que recibes y los que envías
  - 4.4 Observar que el comparador sabe que métodos se han modificado que aparte de un diff normal hace un análisis de la semántica del que compara

---

## TRABAJO A HACER

- 1 El botón tiene que tener nombre internacionalizado (castellano, catalán, y default)
- 2 Hacer los cambios pertinentes y hacer el commit. Asegurarse de que no machacamos nada. Sobretudo al hacer el commit de ficheros de propiedades
- 3 Herramientas: call hierachy, search references in workspace ....
- 4 El botón tiene que hacer algo, cambiar de dirección, cambiar el color en función del sitio donde esté, hacer una filigrana (un ocho, una vuelta, una s...) cambiar la velocidad, poner algun botón en los paneles laterales que haga alguna funcion 'rara'
- 5 SI COMPILA, HACER CHECKOUT, revisar si aun compila, y entonces hacer COMMIT.
- 6 Al final tendríamos que tener unos 15 botones haciendo unas 15 cosas diferentes con la pelota
- 7 Probar "refactorings"
  - 7.1 Cambios de nombres
  - 7.2 Subir / bajar métodos
  - 7.3 Cambios en las firmas
  - 7.4 Ayudas en el código, definimos un atributo y entonces "source-> generate get/set"

Si te gusta puedes probar cosas similares con el Robocode, Coderuler, CodeRally

## COSAS A TENER EN CUENTA

---

Abrir, editar, compilar (no hace falta), ejecutar

- ▶ Problemas
- ▶ generate getter and setter
- ▶ El . despliega el listado de métodos disponibles.
- ▶ Si utilizamos los asistentes CTRL + espacio no nos tenemos que preocupar de los imports.
- ▶ rename in file
- ▶ En las preferencias seleccionar la opción de marcar todas las instancias l'opció de marcar marca totes les instàncies
- ▶ Lo que se pone en el Javadoc aparece en la ayuda que muestra el Eclipse sobre el método.
- ▶ Quien me llama, a quien llamo, y navegar.
  - navegar (f3) te lleva a la función / método
  - Finestra Call - hieracy.
  - Vista Type hieracy
  - De quien vengo, quien me hereda, quien implementa cierto método(Lock view)
- ▶ Compare with each other
- ▶ Suport a JUnit
- ▶ debugar
- ▶ hot code replace (sino cambia la firma de la clase) VM > 1.4 asignacion de valores "a saco"
- ▶ Drop to frame para repetir el método
- ▶ poner TODO en el código para recordar que se tiene que hacer.

Para cualquier duda podeis poner os en contacto con Ermengol Bota en ebota at criptos punt com