

# How to Debug for Backend

May 2019







# Antes de empezar

Necesitas este proyecto de test para seguir la Master Class:

>> Click aquí, definitivamente no virus <<

Haber visto con anterioridad la Master Class de Debugg para front-end puede ayudar:

Presentación debugg frontend

Todo el material ha sido extraído de (creditos):

http://chuwiki.chuidiang.org/index.php?title=Empezando\_con\_el\_debugger\_de\_eclipse

El IDE usado en esta presentación es ECLIPSE, aunque lo que se mostrará existe en todos los IDE.







# **Breakpoint**

## ¿Qué es y cómo ponerlo?

Es un punto de detención de la ejecución que permite visualizar los datos. Para ponerlo se suele dar clic en el lateral del codigo.

- Detiene el runtime
- Permite observar datos en momentos específicos de ejecución
- Permite ver la ejecución línea a línea.

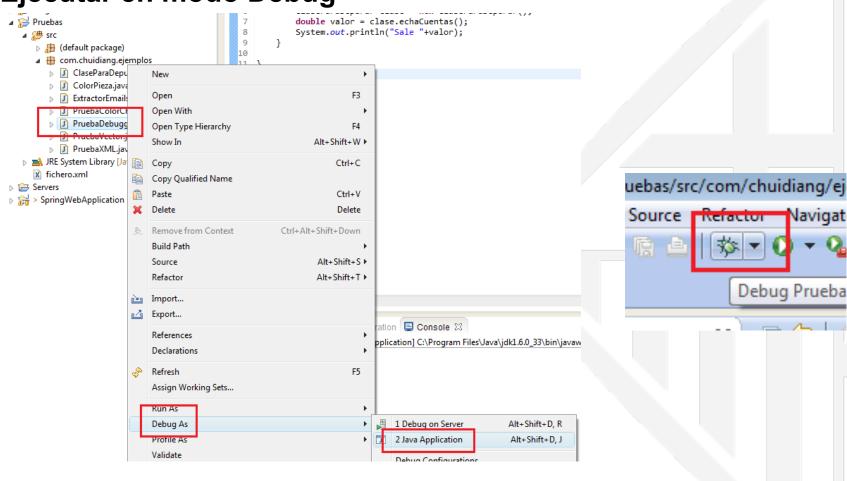
```
PruebaDebugger.java
     package com.chuidiang.ejemplos;
  3⊖ import java.util.LinkedList;
     import java.util.List;
     public class ClaseParaDepurar {
  7
  8
         private double suma;
  9
         private List<Double> lista;
 10
 110
         public ClaseParaDepurar() {
             System.out.println("construyendo clase");
 12
 13
 140
         public double echaCuentas() {
 15
             lista = new LinkedList<Double>();
 16
             for (int i = 0; i<100; i++){
 17
                 lista.add((double)i);
 18
19
     Line breakpoint: ClaseParaDepurar [line: 19] - echaCuentas()
 20
             for (Double valor : lista){
 21
                 suma +=valor;
 22
 23
             return suma;
 24
 25
 26
 27
```





# **Breakpoint**

Ejecutar en modo Debug









# **Breakpoint**

## Navegar en modo debug

Durante el modo debug hay ciertos botones que nos permiten avanzar la ejecución al ritmo que necesitemos.

- Botón para el siguiente BREAKPOINT
- Botón para parar la ejecución modo debug
- Botón para avanzar una línea en la ejecución
- Botón para adentrarse en un método
- Botón para ejecutar hasta salir de un método



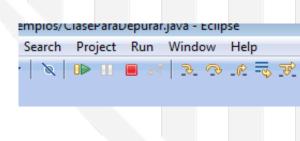


















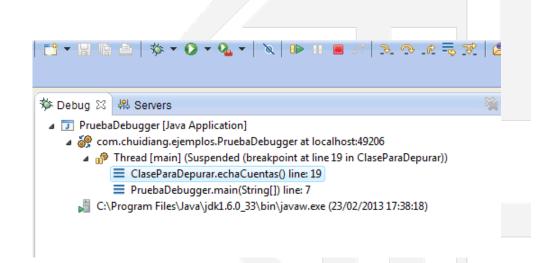
## **CallStack**

#### ¿Qué es?

Podemos ver cada hilo en ejecución por orden

#### Puntos de interés:

 Suele tener similitud con el callStack de errores.









## Visualizar variables

Se pueden ver las variables en el momento de la parada

- Se pueden visualizar en la pestaña
- También se pueden visualizar poniendo el ratón encima en las líneas de código.

```
☑ PruebaDebugger.java
☑ ClaseParaDepurar.java
※
 13
 14⊝
         public double echaCuentas() {
 15
             lista = new LinkedList<Double>();
 16
             for (int i = 0; i<100; i++){
 17
                  lista.add((double)i);
 18
19
             suma = 0.0;
 20
             for (Double valor : lista){
 21
                  suma +=valor;
 22
 23
             return suma;
 24
 25
 26
 27
(x)= Variables 🖂
                                             Value
 Name
                                             ClaseParaDepurar (id=18)
 lista
                                             LinkedList<E> (id=21)
       LinkedList$Entry<E> (id=44)

    modCount

                                             100
         size
                                             100
                                             0.0
      suma
[0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0, 16.
```







## Visualizar variables

```
13
 14⊝
       public double echaCuentas() {
 15
          lista = new LinkedList<Double>();
 16
          for (Int i = 0; i<100; i++){
 17
             lista.add((double)i);
 18
              lista= LinkedList<E> (id=21)
19
                 20
          for
                  ▲ [1]= Double (id=37)
 21
 22
                  ▲ [2]= Double (id=38)
 23
                 24
              [0.0, 44.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0]
 25
 26
```







## **Editar variables**

Se pueden modificar variables para probar otros casos de ejecución

#### Puntos de interés:

 Con tan solo un doble clic basta para editar el valor en la pestaña variables.

```
☑ PruebaDebugger.java
☑ ClaseParaDepurar.java
※
 13
 14⊝
         public double echaCuentas() {
 15
             lista = new LinkedList<Double>();
 16
             for (int i = 0; i<100; i++){
                 lista.add((double)i);
 18
19
             suma = 0.0;
 20
             for (Double valor : lista){
 21
                 suma +=valor;
 22
 23
             return suma;
 24
 25
 26
 27
(x)= Variables 🖂
                                             Value
 Name
                                             ClaseParaDepurar (id=18)
 lista
                                             LinkedList<E> (id=21)
       LinkedList$Entry<E> (id=44)

    modCount

                                             100
         size
                                             100
                                             0.0
      suma
[0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0, 16.
```





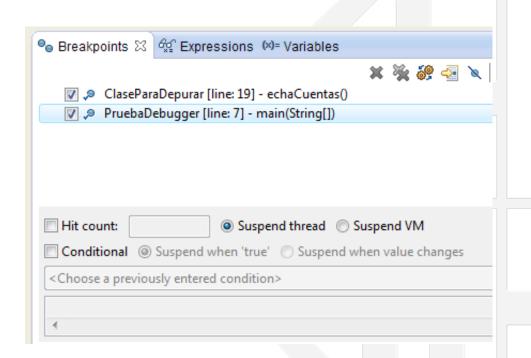


# Breakpoints condicionales.

## Visualizar breakpoints

Disponemos de una ventana para visualizar todos los breakpoints

- Se pueden habilitar y deshabilitar de forma facil
- Se pueden poner condiciones









# Breakpoints condicionales.

### **Definir un breakpoint**

Como un condicional cualquiera en la pestaña breakpoint podemos introducir una condicion de las variables del codigo. en este ejemplo parar cuando i es igual a 50

- Puedes parar la ejecución en momentos específicos
- Incluso parar cuando un valor cambia.

```
☑ PruebaDebugger.java
☑ ClaseParaDepurar.java
☒
                                         private List<Double> lista;
           9
       10
                                           public ClaseParaDepurar() {
       11⊖
       12
                                                            System.out.println("construyendo clase");
       13
       14⊖
                                          public double echaCuentas() {
       15
                                                            lista = new LinkedList<Double>();
       16
                                                            for (int i = 0; i<100; i++){
 17
                                                                              18
                                                                                                                                                                              ① i= 50
   9 19
                                                            suma = 0.0;
       20
                                                             for (Double valor : li
       21
                                                                              suma +=valor;
       22
       23
                                                            return suma;
       24
                                                                                                                                                                50
ClaseParaDepurar [line: 17] [condition of the condition of the conditio
                ClaseParaDepurar [line: 19] - echaCuentas()
      Hit count:
                                                                                                            Suspend thread Suspend VM
    Conditional 
Suspend when 'true'
                                                                                                                                                Suspend when value changes
       < Choose a previously entered condition>
       i==50
```





