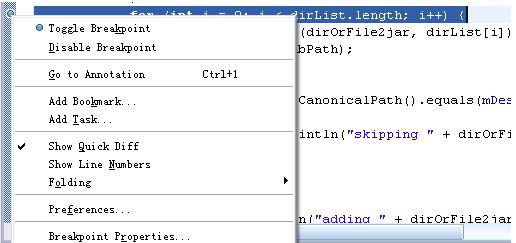
**1、 条件断点**

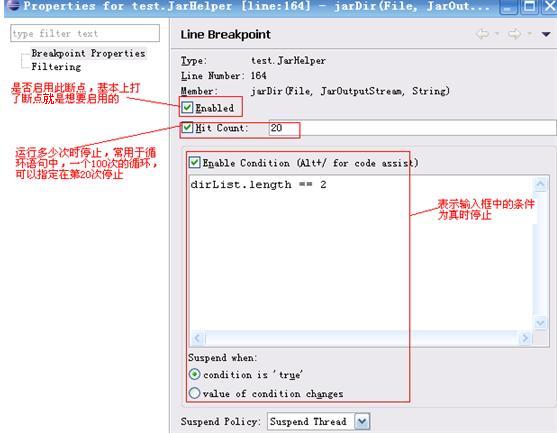
断点大家都比较熟悉，在Eclipse Java 编辑区的行头双击就会得到一个断点，代码会运行到此处时停止。

条件断点，顾名思义就是一个有一定条件的断点，只有满足了用户设置的条件，代码才会在运行到断点处时停止。

在断点处点击鼠标右键，选择最后一个"Breakpoint Properties"

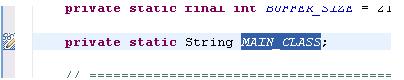


断点的属性界面及各个选项的意思如下图，



**2、 变量断点**

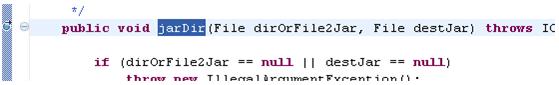
断点不仅能打在语句上，变量也可以接受断点，



上图就是一个变量的打的断点，在变量的值初始化，或是变量值改变时可以停止，当然变量断点上也是可以加条件的，和上面的介绍的条件断点的设置是一样的。

**3、 方法断点**

方法断点就是将断点打在方法的入口处，

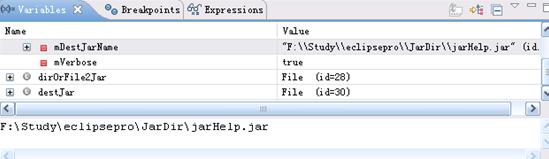


方法断点的特别之处在于它可以打在 JDK的源码里，由于 JDK 在编译时去掉了调试信息，所以普通断点是不能打到里面的，但是方法断点却可以，可以通过这种方法查看方法的调用栈。

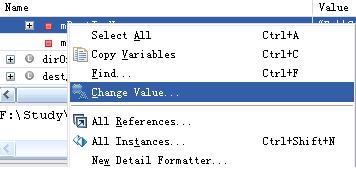
**4、 改变变量值**

代码停在了断点处，但是传过来的值不正确，如何修改一下变量值保证代码继续走正确的流程，或是说有一个异常分支老是进不去，能不能调试时改一下条件，看一下异常分支代码是否正确？

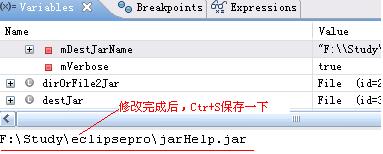
在Debug 视图的 Variables 小窗口中，我们可以看到 mDestJarName 变量的值为 "F:\Study\eclipsepro\JarDir\jarHelp.jar "



我们可以在变量上右键，选择"Change Value..." 在弹出的对话框中修改变量的值，



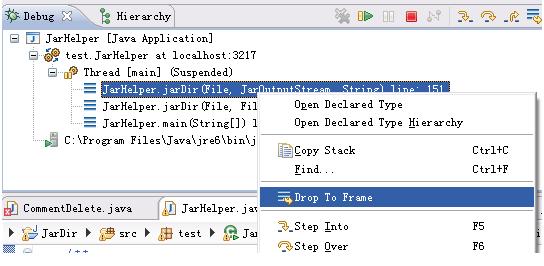
或是在下面的值查看窗口中修改，保用Ctr+S 保存后，变量值就会变成修改后的新值了。



**5、 重新调试**

这种调试的回退不是万能的，只能在当前线程的栈帧中回退，也就说最多只能退回到当前线程的调用的开始处。

回退时，请在需要回退的线程方法上点右键，选择 "Drop to Frame"



**6、 远程调试**

用于调试不在本机上的程序，有两种方式，

1、本机作为客户端

2、本机作为服务端

使用远程调试的前提是服务器端和客户端的代码是一致的。

**本机作为客户端**

本机作客户端比较常用，需要在远端的服务器上的java程序在启动时打开远程调试开关，

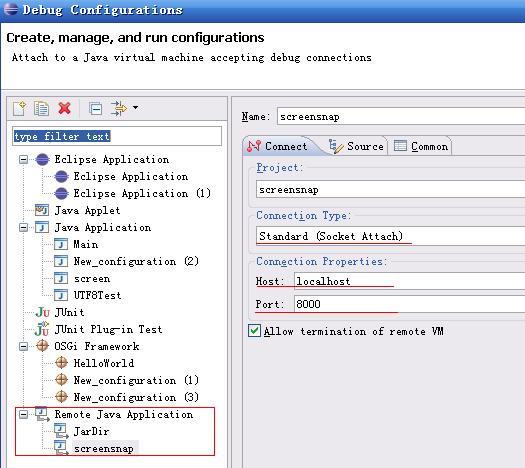
服务器端需要加上虚拟机参数

1.5以前版本（1.5以后也可用）：【-Xdebug -Xrunjdwp:transport=dt\_socket,server=y,address=8000 】

1.5及以上版本：【 -agentlib:jdwp=transport=dt\_socket,server=y,address=8000】

F:\Study\eclipsepro\screensnap>java -Xdebug -Xrunjdwp:transport=dt\_socket,server=y,address=8000 -jar screensnap3.jar

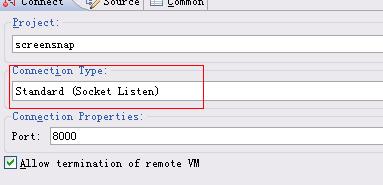
连接时远程服务器时，需要在Eclipse中新建一个远程调试程序



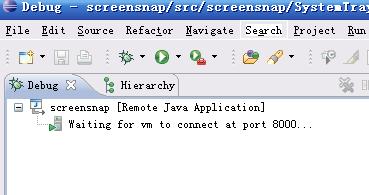
这里有一个小地方需注意，连接上的时候貌似不能自动切换到Debug视图，不要以为本机的调试程序没有连接到服务器端。

**本机作为服务端**

同本机作为客户端相比，只需要修改一下“Connection Type”



这时Eclipse会进入到等待连接的状态



连接程序使用如下参数即可连接本机服务器，IP地址请用实现IP替换~~

【-agentlib:jdwp=transport=dt\_socket,suspend=y,address=127.0.0.1:8000】

F:\Study\eclipsepro\screensnap>java -agentlib:jdwp=transport=dt\_socket,suspend=y,address=127.0.0.1:8000 -jar screensnap3.jar

远程调试时本地的代码修改可同步到远程，但不会写到远程的文件里，也就是说本地修改会在下次启动远程程序时就没有了，不会影响到下次使用时的远程代码。

有关远程调试更详细点的介绍请参考[【使用 Eclipse 远程调试 Java 应用程序】](http://www.ibm.com/developerworks/cn/opensource/os-eclipse-javadebug/)

好像漏了一个断点，异常断点，补一下。

**7、异常断点**

经常遇见一些异常，然后程序就退出来了，要找到异常发生的地方就比较难了，还好可以打一个异常断点，



上图中我们增加了一个NullPointException的异常断点，当异常发生时，代码会停在异常发生处，定位问题时应该比较有帮助。