### 一、Debug开篇

②打上/取消断点快捷键Ctrl+F8（maccommand+F8）

③Debug窗口：访问请求达到第一个断点后，会自动激活Debug窗口。

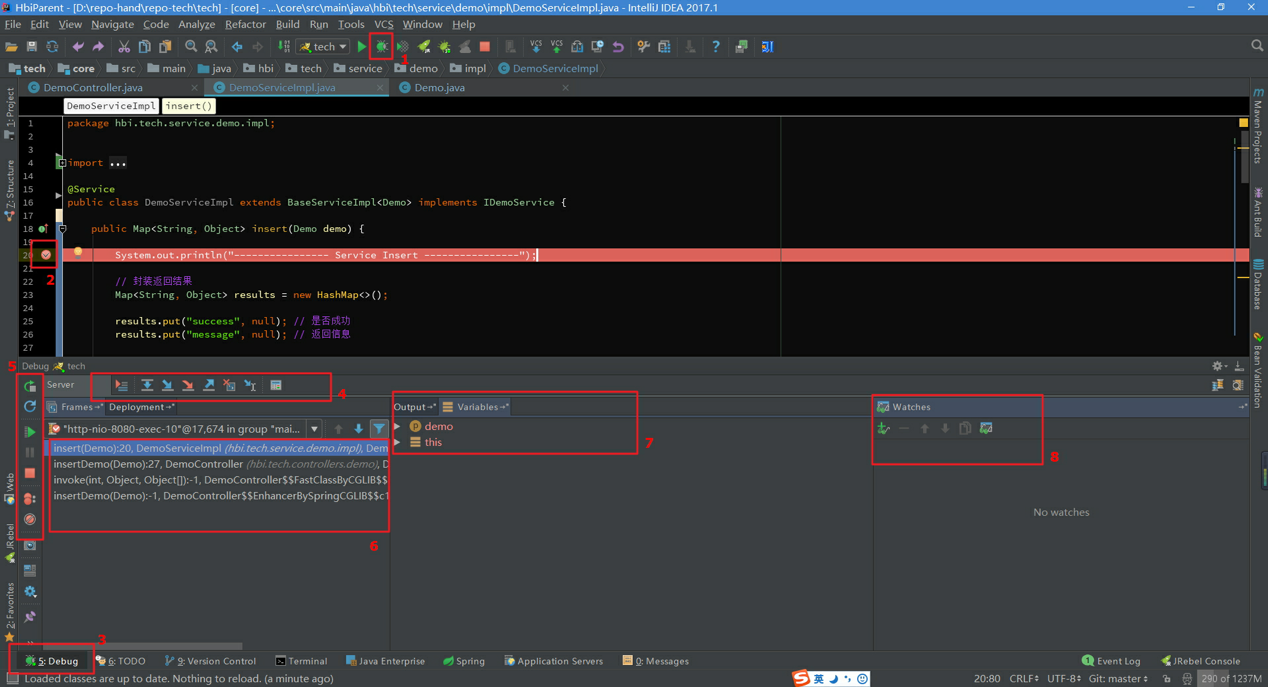
④调试按钮：8个按钮，调试的主要功能就对应着这几个按钮，鼠标悬停在按钮上可以查看对应的快捷键。在菜单栏Run里可以找到同样的对应的功能。

⑤服务按钮：可以在此处关闭/启动服务，设置断点等。

⑥方法调用栈：这里显示了该线程调试所经过的所有方法，勾选右上角的【ShowAllFrames】按钮，就不会显示其他类库的方法了，否则这里会有一大堆方法。

⑦Variables：在变量区可以查看当前断点之前的当前方法内的变量。

⑧Watches：查看变量，可以将Variables区中的变量拖到Watches中查看。



### 二、基本用法&快捷键

1、图中④的一组按钮，共8个：



>ShowExecutionPoint(Alt+F10)：如果光标在其他行或其他页面，点击这个按钮跳转到当前代码执行的行。

>StepOver(F8)：步过，一行一行地往下走，如果当前行上有方法不会进入方法。

>StepInto(F7)：步入，如果当前行有方法，可以进入方法内部，一般用于进入自定义方法内，不会进入官方类库的方法。

>ForceStepInto(Alt+Shift+F7)：强制步入，能进入任何方法，查看底层源码的时候可以用这个进入官方类库的方法。

>StepOut(Shift+F8)：步出，从步入的方法内退出到方法调用处，此时方法已执行完毕，只是还没有完成赋值。

>DropFrame(默认无)：回退断点。

>RuntoCursor(Alt+F9)：运行到光标处，你可以将光标定位到你需要查看的那一行，然后使用这个功能，代码会运行至光标行，而不需要打断点。

>EvaluateExpression(Alt+F9)：计算表达式。

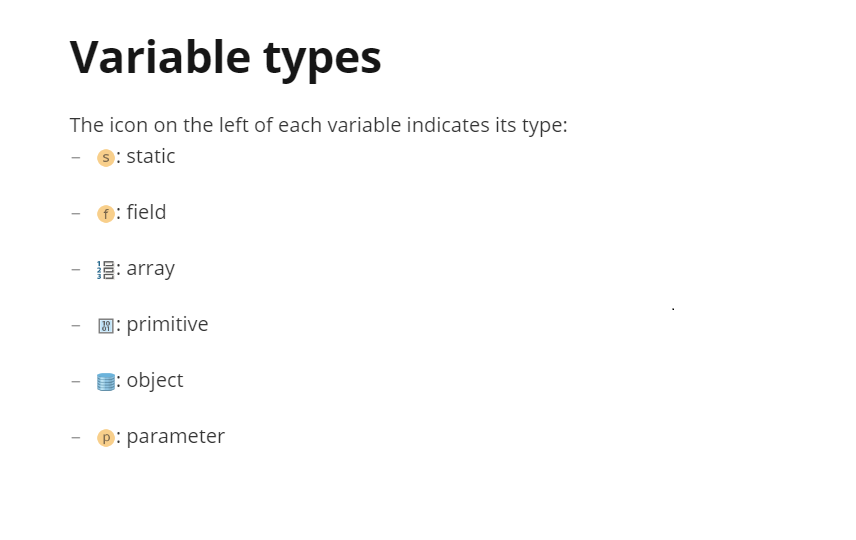
2、图中⑤的一组按钮，共6个：



>ViewBreakpoints(Ctrl+Shift+F8)：查看所有断点。

>MuteBreakpoints：哑的断点，选择这个后，所有断点变为灰色，断点失效，按F9则可以直接运行完程序。再次点击，断点变为红色，有效。

### 三、变量查看



查看变量的几个地方：

1、参数所在行后面会显示当前变量的值。

2、光标悬停到参数上，显示当前变量信息，可以点击打开详情。

3、在Variables里查看，这里显示当前方法里的所有变量。

4、在Watches里，点击NewWatch，输入需要查看的变量。或者从Variables里拖拽。

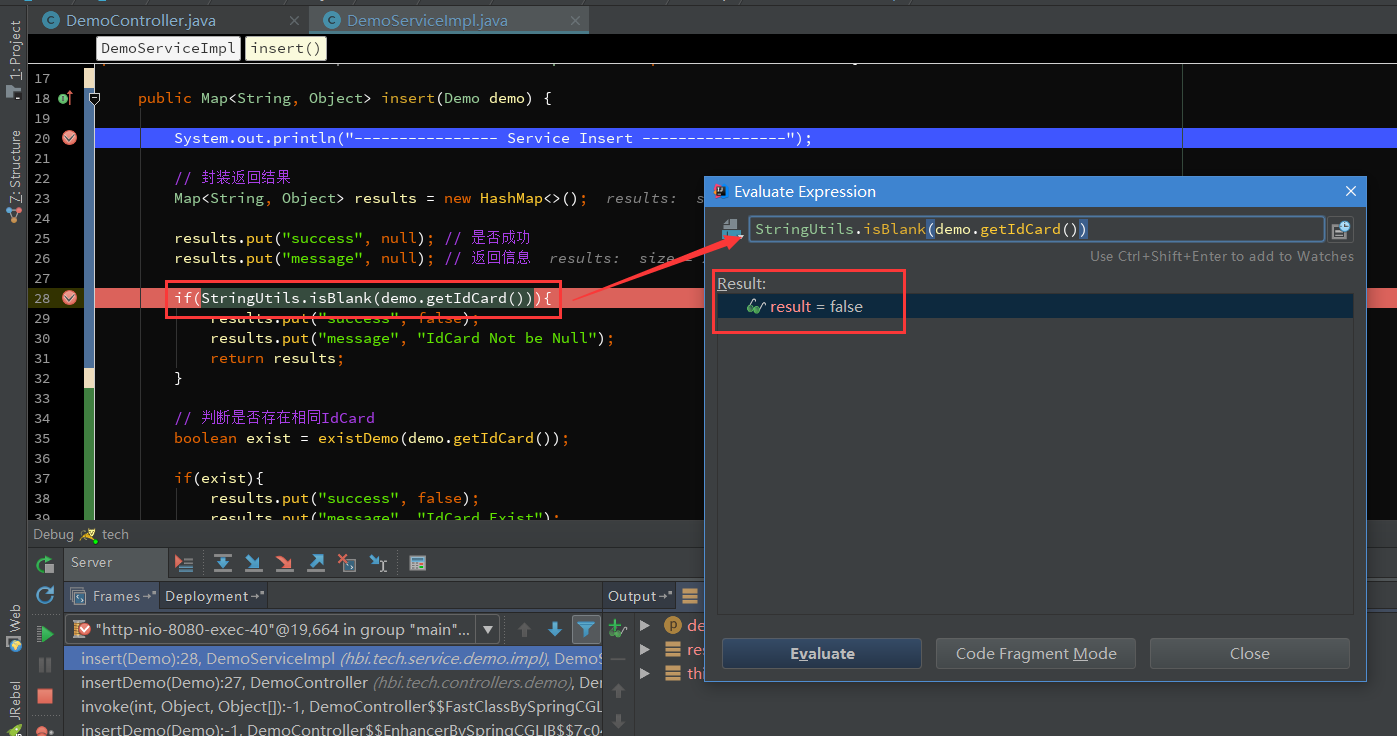
### 四、计算表达式

可以使用这个操作在调试的过程中计算某个表达式的值，不用再去打印信息。

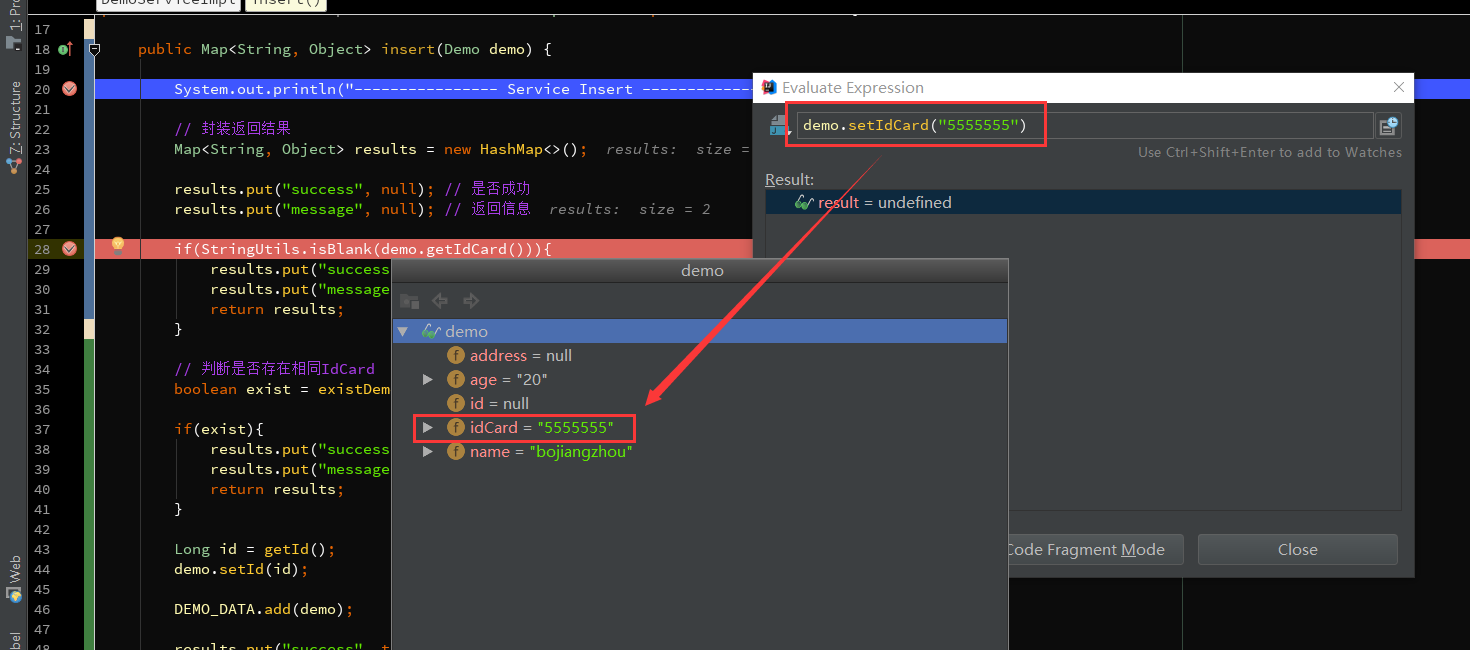


1、按Alt+F8或按钮，或者，你可以选中某个表达式再Alt+F8，弹出计算表达式的窗口，如下，回车或点击Evaluate计算表达式的值。

这个表达式不仅可以是一般变量或参数，也可以是方法，当你的一行代码中调用了几个方法时，就可以通过这种方式查看查看某个方法的返回值。



2、设置变量，在计算表达式的框里，可以改变变量的值，这样有时候就能很方便我们去调试各种值的情况了不是。



### 五、智能步入

想想，一行代码里有好几个方法，怎么只选择某一个方法进入。之前提到过使用StepInto(Alt+F7)或者ForceStepInto(Alt+Shift+F7)进入到方法内部，但这两个操作会根据方法调用顺序依次进入，这比较麻烦。

那么智能步入就很方便了，智能步入，这个功能在Run里可以看到，SmartStepInto(Shift+F7)，如图

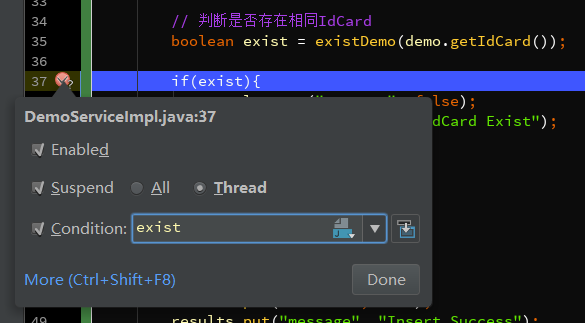


### 六、断点条件设置

设置断点条件，在满足条件时，才停在断点处，否则直接运行。

通常，当我们在遍历一个比较大的集合或数组时，在循环内设置了一个断点，难道我们要一个一个去看变量的值？那肯定很累，说不定你还错过这个值得重新来一次。

1、在断点上右键直接设置当前断点的条件，如图6.1，我设置exist为true时断点才生效。

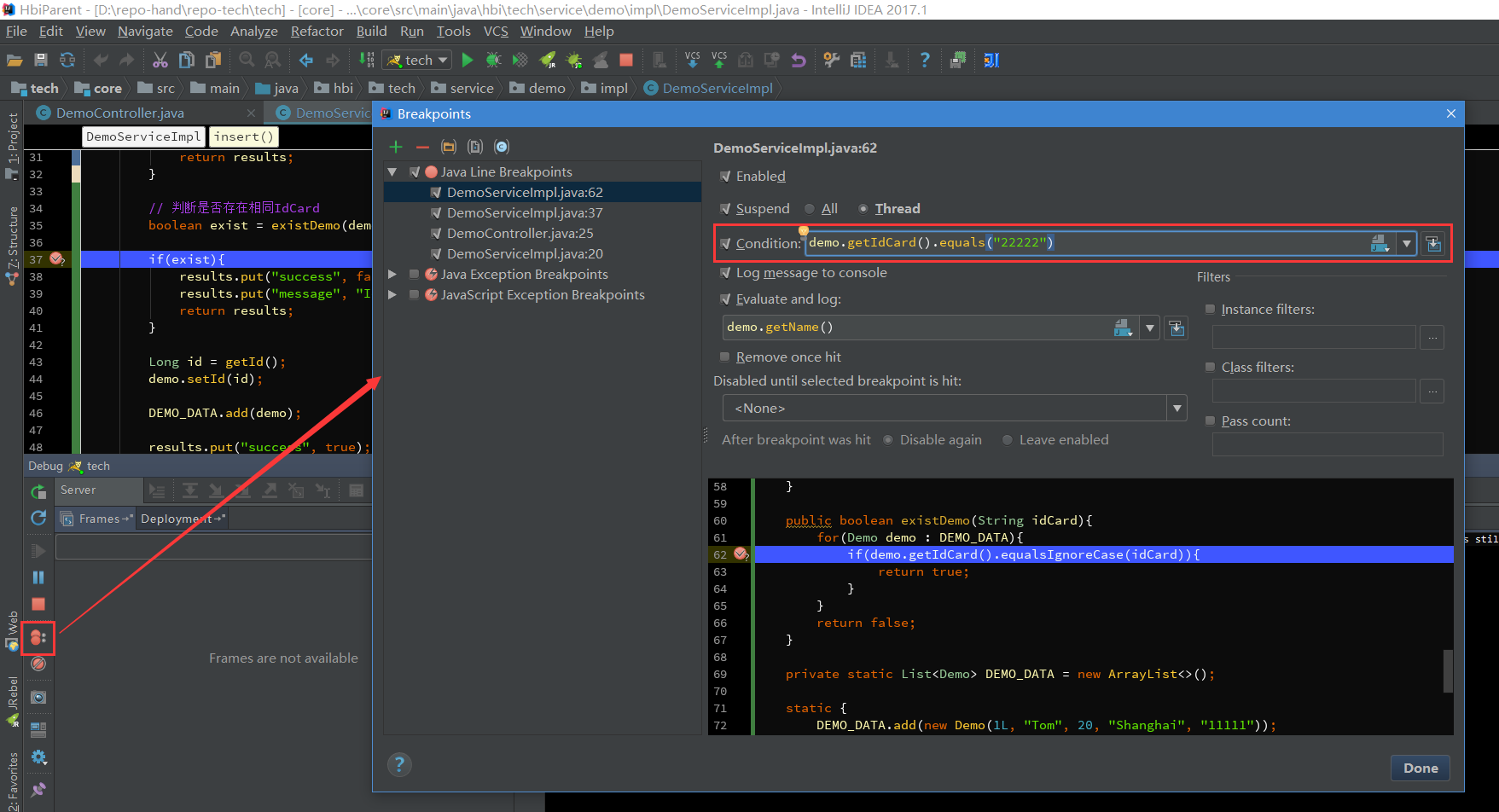


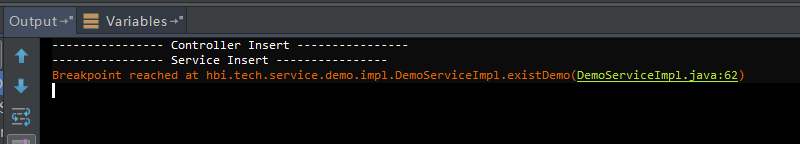
2、点击View Breakpoints(Ctrl+Shift+F8)，查看所有断点。

Java Line Breakpoints显示了所有的断点，在右边勾选Condition，设置断点的条件。

勾选Log message to console，则会将当前断点行输出到控制台，如图

勾选Evaluate and log，可以在执行这行代码是计算表达式的值，并将结果输出到控制台。



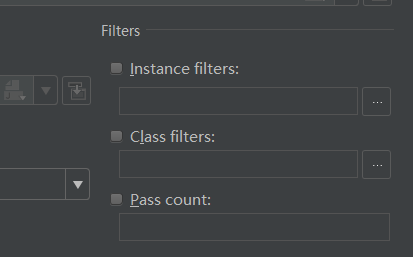


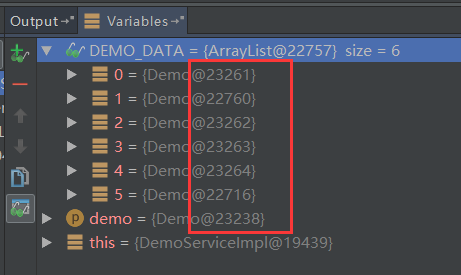
3、再说说右边的Filters过滤，这些一般情况下不常用，简单说下意思。

Instance filters：实例过滤，输入实例ID(如图6.5中的实例ID)，但是我这里没有成功，不知道什么原因，知道的朋友留个言。

Class filters：类过滤，根据类名过滤，同样没有成功....

Pass count：用于循环中，如果断点在循环中，可以设置该值，循环多少次后停在断点处，之后的循环都会停在断点处。

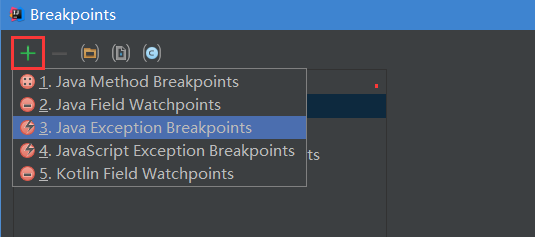


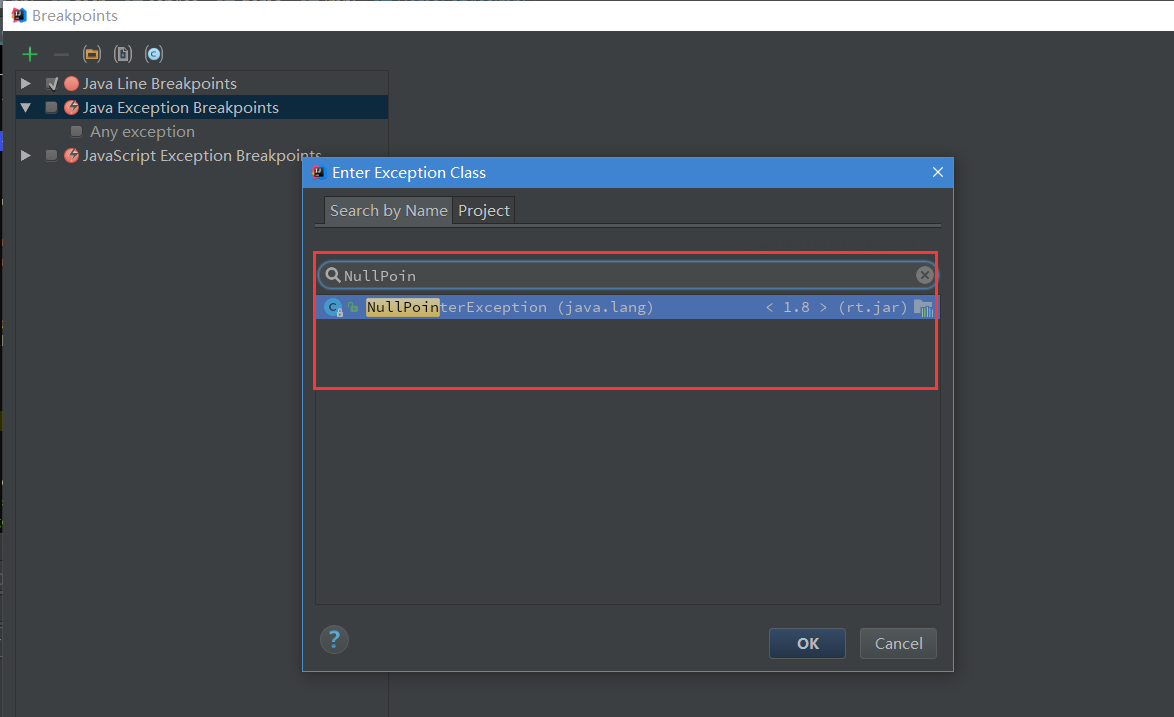


4、异常断点，通过设置异常断点，在程序中出现需要拦截的异常时，会自动定位到异常行。

如图，点击+号添加Java Exception Breakpoints，添加异常断点。然后输入需要断点的异常类，如图，之后可以在Java Exception Breakpoints里看到添加的异常断点。

这里添加了一个NullPointerException异常断点，如图6.8，出现空指针异常后，自动定位在空指针异常行。



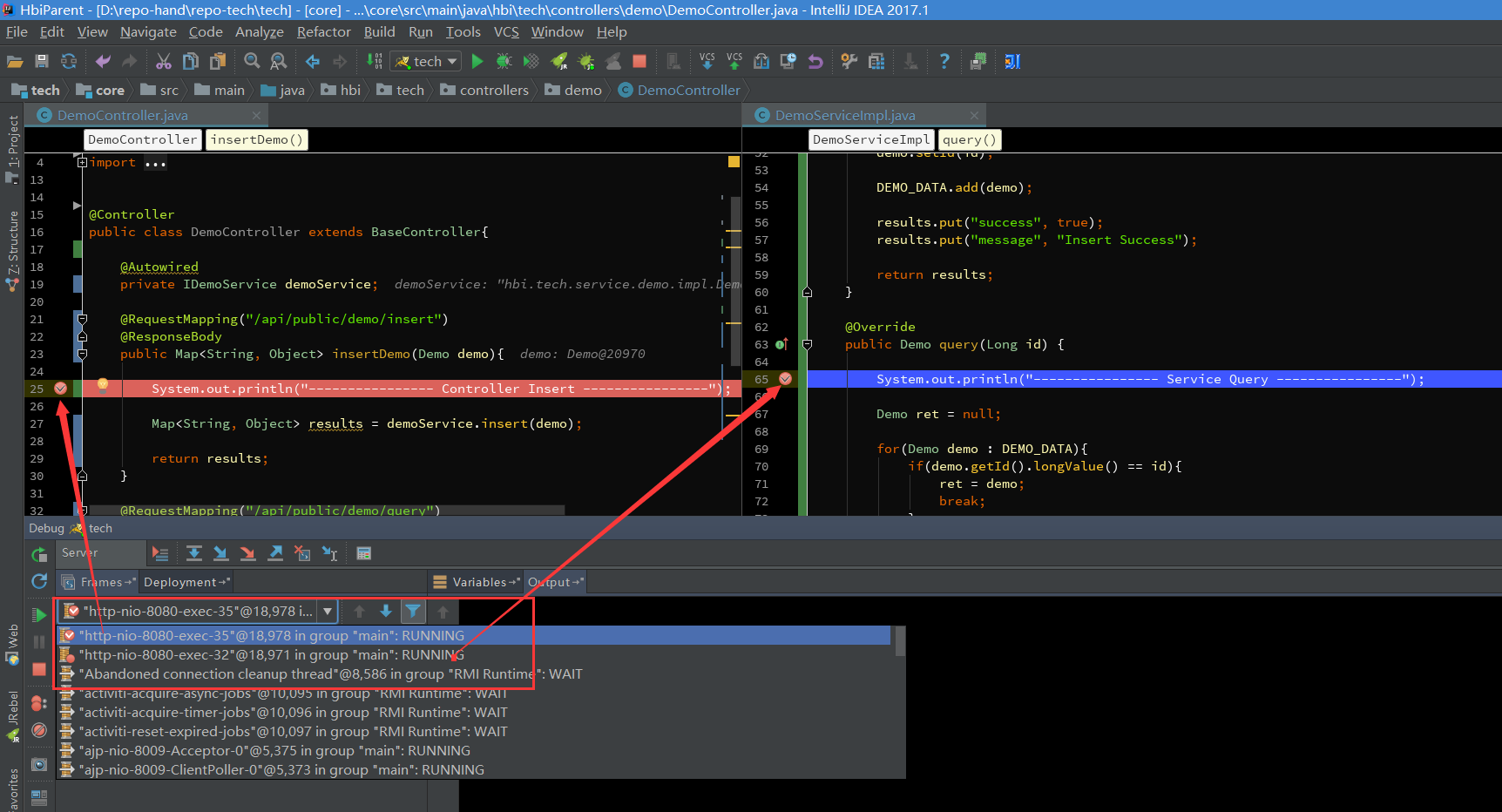


### 七、多线程调试

一般情况下我们调试的时候是在一个线程中的，一步一步往下走，但有时候会发现在Debug的时候，想发起另外一个请求都无法进行了？

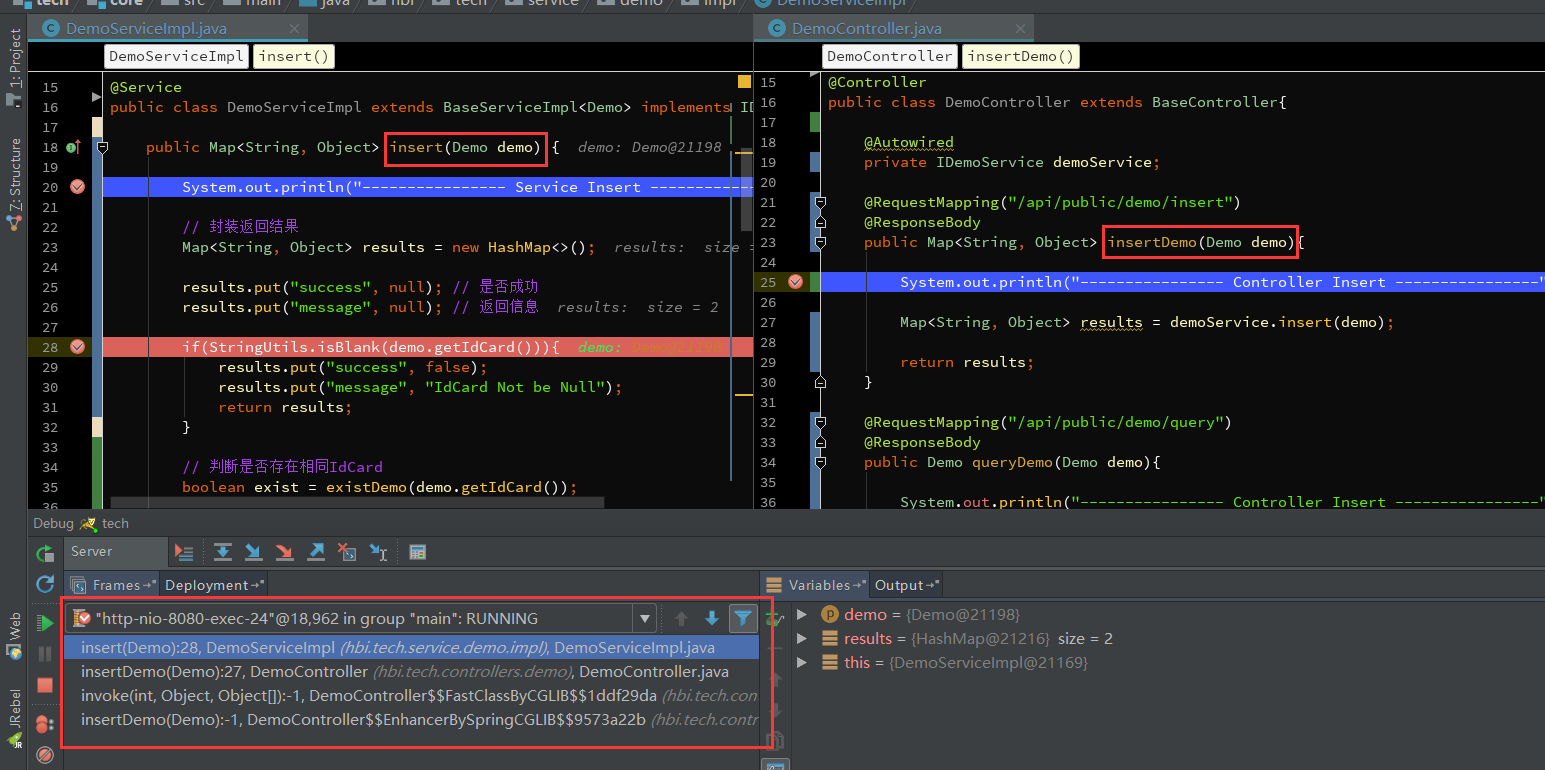
因为IDEA在Debug时默认阻塞级别是ALL，会阻塞其它线程，只有在当前线程走完时才会走其他线程。可以在View Breakpoints里选择Thread，如图，然后点击Make Default设置为默认选项。

线程切换，在图中Frames的下拉列表里，可以切换当前的线程，如下我这里有两个Debug的线程，切换另外一个则进入另一个Debug的线程。



### 八、回退断点

1、首先认识下这个方法调用栈，如图，首先请求进入DemoController的insertDemo方法，然后调用insert方法，其他的invoke我们且先不管，最上面的方法是当前断点所在的方法。



### 2、断点回退

在调试的时候，想要重新走一下流程而不用再次发起一个请求？

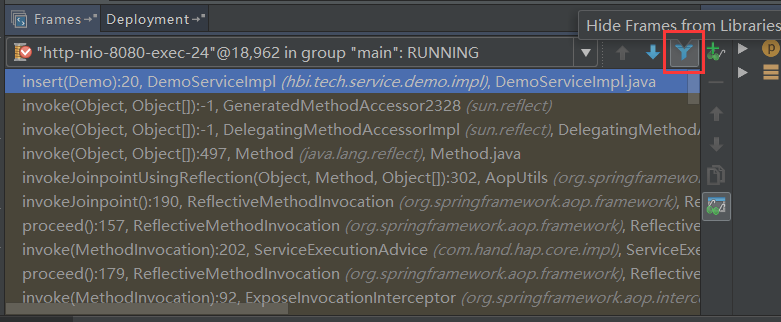
所谓的断点回退，其实就是回退到上一个方法调用的开始处，在IDEA里测试无法一行一行地回退或回到上一个断点处，而是回到上一个方法。

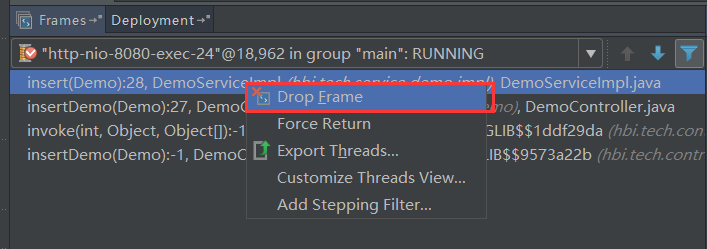
两种回退的方式：

1）Drop Frame按钮，按调用的方法逐步回退，包括三方类库的其它方法（取消Show All Frames按钮会显示三方类库的方法）。

2）在调用栈方法上选择要回退的方法，右键选择Drop Frame，回退到该方法的上一个方法调用处，此时再按F9(Resume Program)，可以看到程序进入到该方法的断点处了。

有一点需要注意，断点回退只能重新走一下流程，之前的某些参数/数据的状态已经改变了的是无法回退到之前的状态的，如对象、集合、更新了数据库数据等等。





### 九、中断Debug

想要在Debug的时候，中断请求，不再走剩余的流程。

可以通过Force Return，即强制返回来避免后续的流程，如图

