

目录

第1章 编码

01 开篇词：为什么学习本专栏 [已学完](#)

02 Integer缓存问题分析

03 Java序列化引发的血案

04 学习浅拷贝和深拷贝的正确姿势 [已学完](#)

05 分层领域模型使用解读 [已学完](#)

06 Java属性映射的正确姿势 [已学完](#)

07 过期类、属性、接口的正确处理姿势 [已学完](#)

08 空指针引发的血案 [已学完](#)

09 当switch遇到空指针

10 枚举类的正确学习方式

11 ArrayList的subList和Arrays的asList学习

12 添加注释的正确姿势

13 你真得了解可变参数吗？

14 集合去重的正确姿势

15 学习线程池的正确姿势

16 虚拟机退出时机问题研究

17 如何解决条件语句的多层嵌套问题？

加餐1：工欲善其事必先利其器

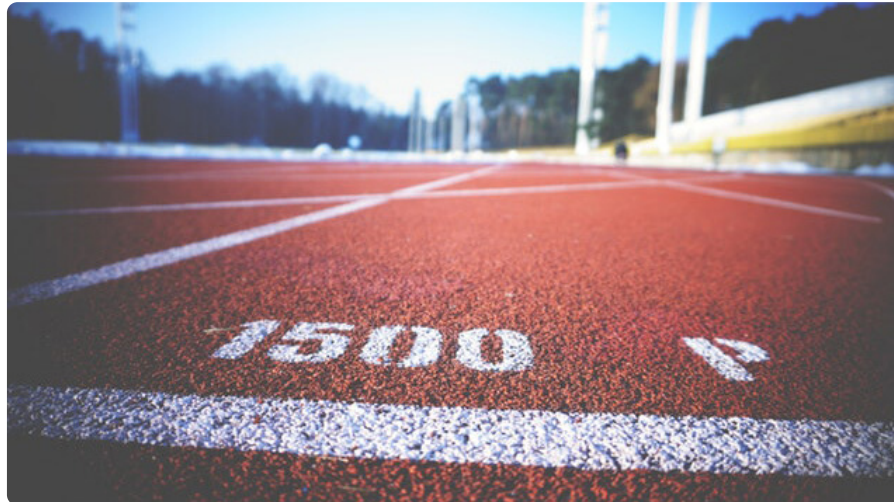
第2章 异常日志

18 一些异常处理建议

19 日志学习和使用的正确姿势

24 代码调试的正确姿势

更新时间：2019-12-11 09:49:34



“

勤能补拙是良训，一分辛劳一分才。

——华罗庚

”

1. 前言

《手册》对代码调试介绍较少，其中第 37 页 SQL 语句章节有如下描述 [1](#)：

【强制】禁止使用存储过程，存储过程难以调试和扩展，更没有移植性。

由此可见，可调试也是 Java 程序员编码要考虑的重要一环。

可以说，代码调试是 java 程序员的必备技能之一。

但是很多 Java 初学者和工作一两年的程序员仍然存在使用“打印语句”来代替调试的现象。还有很多 Java 程序员只了解最基本的调试方法，并没有主动学习和掌握高级的调试技巧。

本节将在 IDEA 中展示常见的调试方法和高级的调试技巧。

2. 调试的好处

调试和日志是排查问题的两个主要手段。

如果没有调试功能，很多问题的排查更多地将依赖日志。

但是日志无法直观地了解代码运行的状态，无法实时地观察待调试对象的各种属性值等。

目录

第1章 编码

01 开篇词：为什么学习本专栏 [已学完](#)

02 Integer缓存问题分析

03 Java序列化引发的血案

04 学习浅拷贝和深拷贝的正确姿势 [已学完](#)

05 分层领域模型使用解读 [已学完](#)

06 Java属性映射的正确姿势 [已学完](#)

07 过期类、属性、接口的正确处理姿势 [已学完](#)

08 空指针引发的血案 [已学完](#)

09 当switch遇到空指针

10 枚举类的正确学习方式

11 ArrayList的subList和Arrays的asList学习

12 添加注释的正确姿势

13 你真得了解可变参数吗？

14 集合去重的正确姿势

15 学习线程池的正确姿势

16 虚拟机退出时机问题研究

17 如何解决条件语句的多层嵌套问题？

加餐1：工欲善其事必先利其器

第2章 异常日志

18 一些异常处理建议

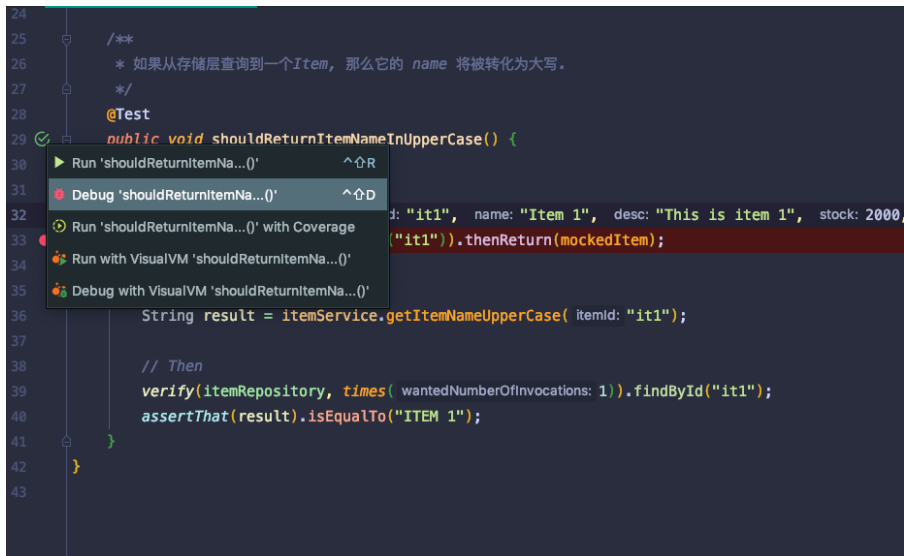
19 日志学习和使用的正确姿势

3. 调试的基本方法

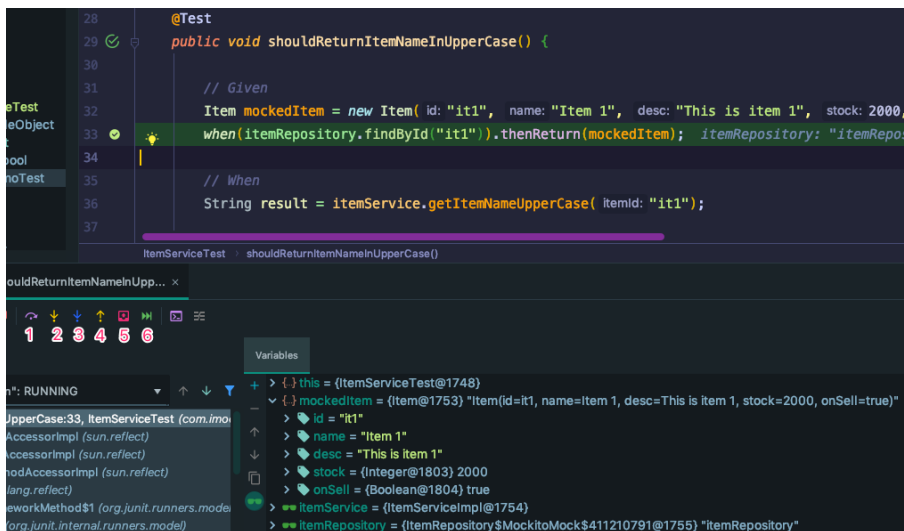
调试的基本步骤：

1. 设置断点
2. 调试模式运行
3. 单步调试

如图所示，在单元测试类的 33 行设置断点，然后在测试类或函数上执行 debug，则程序执行到断点时会暂停。



此时可以看到所有的变量：



常见的调试功能按钮如上图所示。

- 1 表示 Step Over 即跳过，执行到下一行；
- 2 表示 Step Into 即步入，可以进入自定义的函数；

目录

第1章 编码

01 开篇词：为什么学习本专栏 [已学完](#)

02 Integer缓存问题分析

03 Java序列化引发的血案

04 学习浅拷贝和深拷贝的正确方式 [已学完](#)

05 分层领域模型使用解读 [已学完](#)

06 Java属性映射的正确姿势 [已学完](#)

07 过期类、属性、接口的正确处理姿势 [已学完](#)

08 空指针引发的血案 [已学完](#)

09 当switch遇到空指针

10 枚举类的正确学习方式

11 ArrayList的subList和Arrays的asList学习

12 添加注释的正确姿势

13 你真得了解可变参数吗？

14 集合去重的正确姿势

15 学习线程池的正确姿势

16 虚拟机退出时机问题研究

17 如何解决条件语句的多层嵌套问题？

加餐1：工欲善其事必先利其器

第2章 异常日志

18 一些异常处理建议

19 日志学习和使用的正确姿势

4 表示 Step Out 即跳出，如果当前调试的方法没问题，可以使用此功能跳出当前函数；

5 表示 Drop frame 即移除帧，相当于回退到上一级；

6 表示 Run to Cursor 即执行到鼠标所在的代码行数。

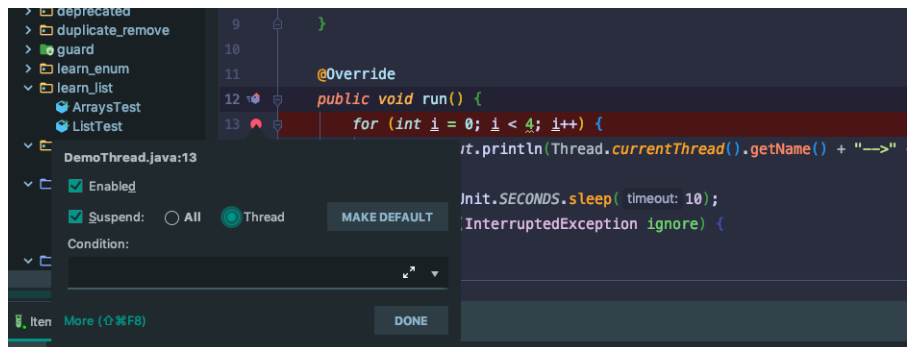
其中 1、2、3、4、6 这 5 个功能，以及“variables（变量区）”初学者用的最多。

通常设置断点后，通过单步观察运行步骤，通过变量区观察“当前”的数据状况，来学习源码或者排查错误的原因。

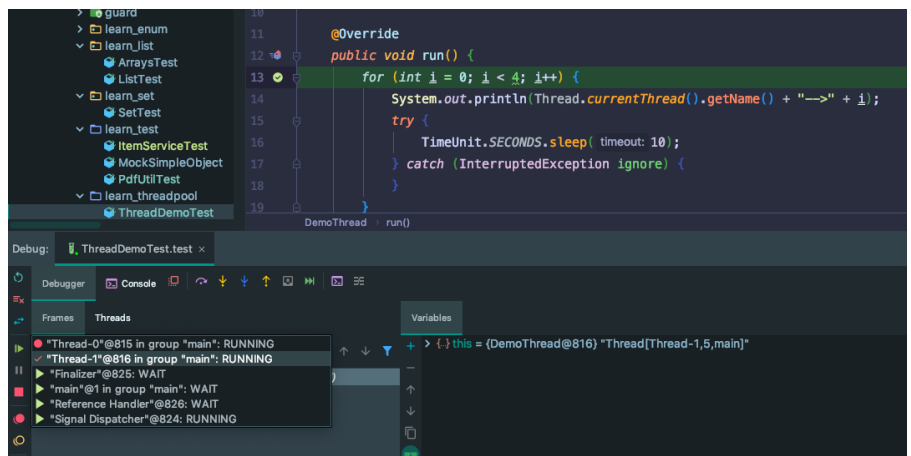
4. 调试的高级技巧

4.1 多线程调试

设置断点时，在断点上右键可以选择断点的模式，选择"Thread"模式，可以开启多线程调试。



可以将一个线程断下来，通过“Frames”选项卡切换到不同线程线程，控制不同线程的运行。



该调试技巧在模拟线程安全问题时非常方便。

4.2 条件断点

和多线程调试类似，我们还可以对断点设置条件，只有满足设置的条件才会生效。

目录

第1章 编码

01 开篇词：为什么学习本专栏 已学完

02 Integer缓存问题分析

03 Java序列化引发的血案

04 学习浅拷贝和深拷贝的正确姿势 已学完

05 分层领域模型使用解读 已学完

06 Java属性映射的正确姿势 已学完

07 过期类、属性、接口的正确处理姿势 已学完

08 空指针引发的血案 已学完

09 当switch遇到空指针

10 枚举类的正确学习方式

11 ArrayList的subList和Arrays的asList学习

12 添加注释的正确姿势

13 你真得了解可变参数吗？

14 集合去重的正确姿势

15 学习线程池的正确姿势

16 虚拟机退出时机问题研究

17 如何解决条件语句的多层嵌套问题？

加餐1：工欲善其事必先利其器

第2章 异常日志

18 一些异常处理建议

19 日志学习和使用的正确姿势



该功能在测试环境中非常有用。

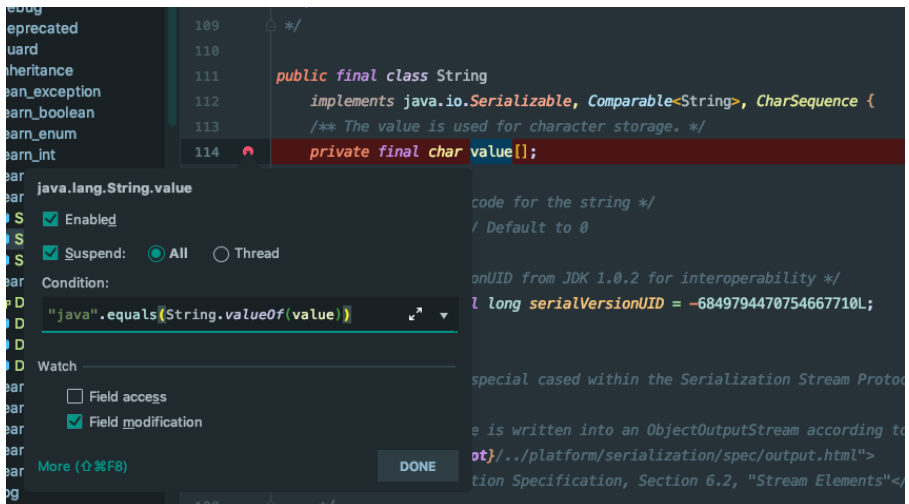
比如你提供视频的转码功能作为二方库给其他团队使用，此时代码发布到测试环境，如果设置普通断点，那么所有的请求都会被暂停，影响其他功能的调试。

此时就可以设置条件断点，将某个待测试的视频 ID 或者业务方 ID 等关键标识作为断点的条件，就不会相互影响。

如果我们想对某个成员变量修改的地方打断点，但是修改的地方特别多怎么办？

难道每个地方都要打断点？

如下图所示，我们可以在属性上加断点，选择在属性访问或修改时断点，还可以加上断点生效的条件。



4.3 “后悔药”

在基本调试方法部分讲到，按钮 5 表示 Drop frame 即移除帧，相当于回退到上一级，这给我们提供了“后悔药”。

当我们调试某个问题时，一不小心走过了，往往会重新运行调试，非常浪费时间，此时可以通过该功能实现“回退”。

比如我们在 33 行设置断点，通过 step over 走到了第 36 行。

目录

第1章 编码

01 开篇词：为什么学习本专栏 已学完

02 Integer缓存问题分析

03 Java序列化引发的血案

04 学习浅拷贝和深拷贝的正确方式 已学完

05 分层领域模型使用解读 已学完

06 Java属性映射的正确姿势 已学完

07 过期类、属性、接口的正确处理姿势 已学完

08 空指针引发的血案 已学完

09 当switch遇到空指针

10 枚举类的正确学习方式

11 ArrayList的subList和Arrays的asList学习

12 添加注释的正确姿势

13 你真得了解可变参数吗？

14 集合去重的正确姿势

15 学习线程池的正确姿势

16 虚拟机退出时机问题研究

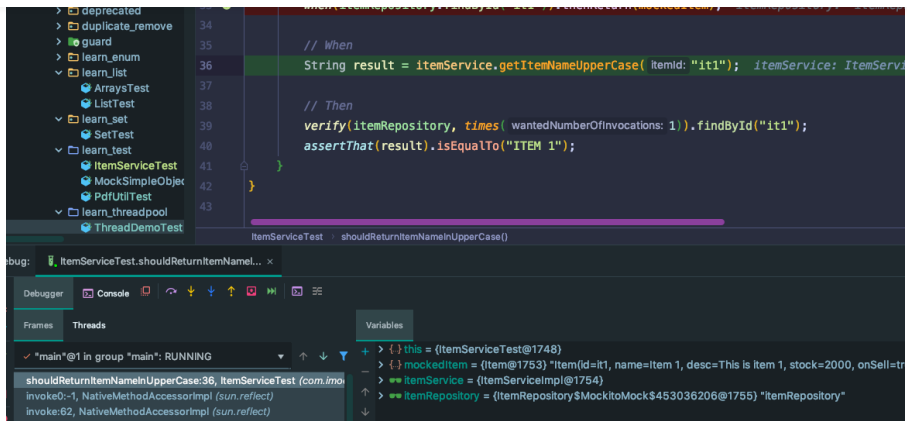
17 如何解决条件语句的多层嵌套问题？

加餐1：工欲善其事必先利其器

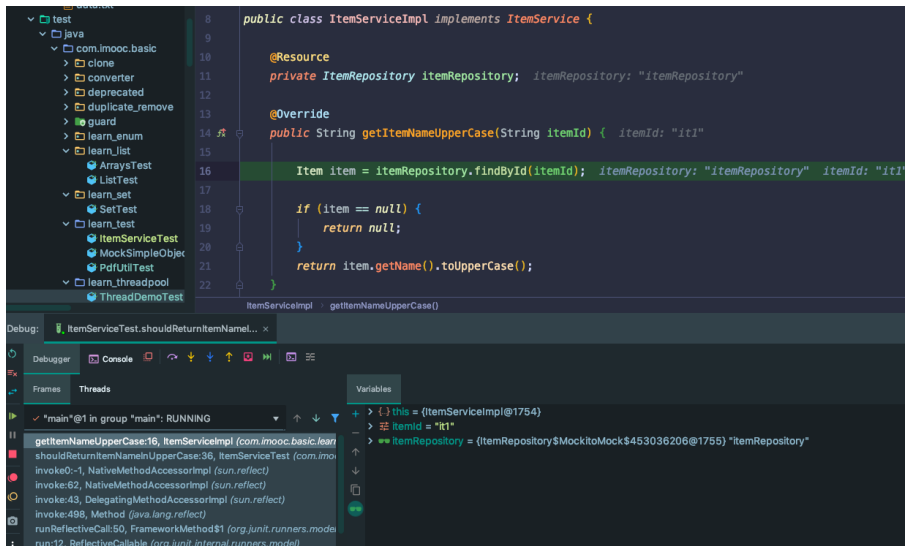
第2章 异常日志

18 一些异常处理建议

19 日志学习和使用的正确姿势



然后通过 step into 来走到了 `ItemServiceImpl` 的第 16 行，如果我们想回退到上一层，直接使用 drop frame 功能即可回退到上图状态重新调试。



4.4 “偷天换日”

我们实际调试代码时，会有这样的场景，调用的参数传错了。修改参数重新运行？

不需要，我们可以在调试过程中对调试对象的值进行动态修改。

如程序运行到 39 行时 result 的值为 “ITEM 1”，如果我们想对其进行修改。

目录

第1章 编码

01 开篇词：为什么学习本专栏 已学完

02 Integer缓存问题分析

03 Java序列化引发的血案

04 学习浅拷贝和深拷贝的正确方式 已学完

05 分层领域模型使用解读 已学完

06 Java属性映射的正确姿势 已学完

07 过期类、属性、接口的正确处理姿势 已学完

08 空指针引发的血案 已学完

09 当switch遇到空指针

10 枚举类的正确学习方式

11 ArrayList的subList和Arrays的asList学习

12 添加注释的正确姿势

13 你真得了解可变参数吗？

14 集合去重的正确姿势

15 学习线程池的正确姿势

16 虚拟机退出时机问题研究

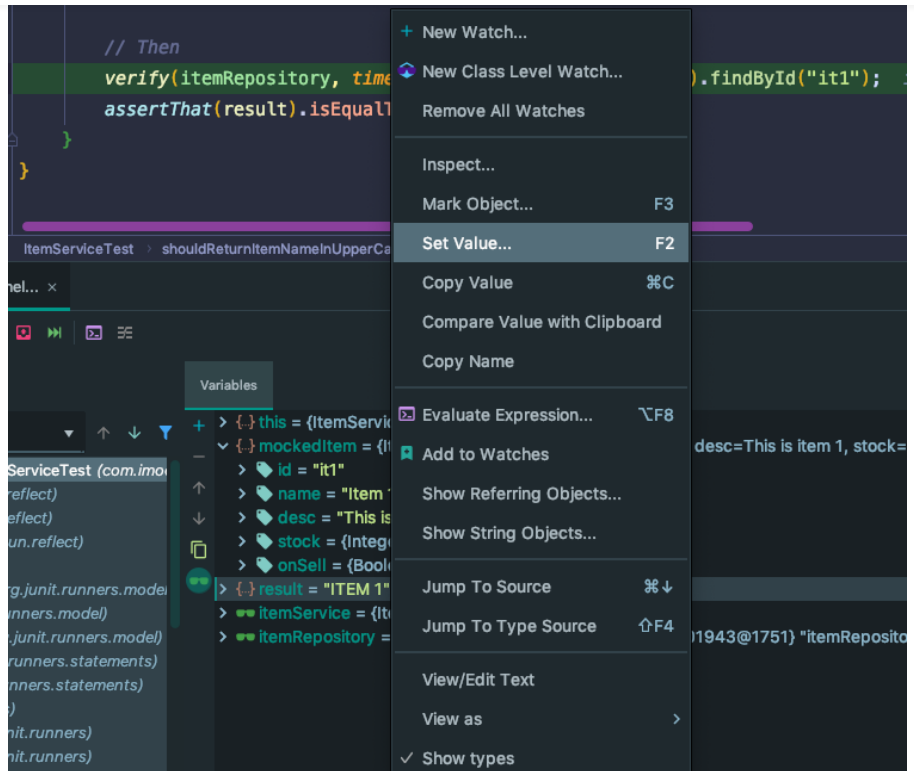
17 如何解决条件语句的多层嵌套问题？

加餐1：工欲善其事必先利其器

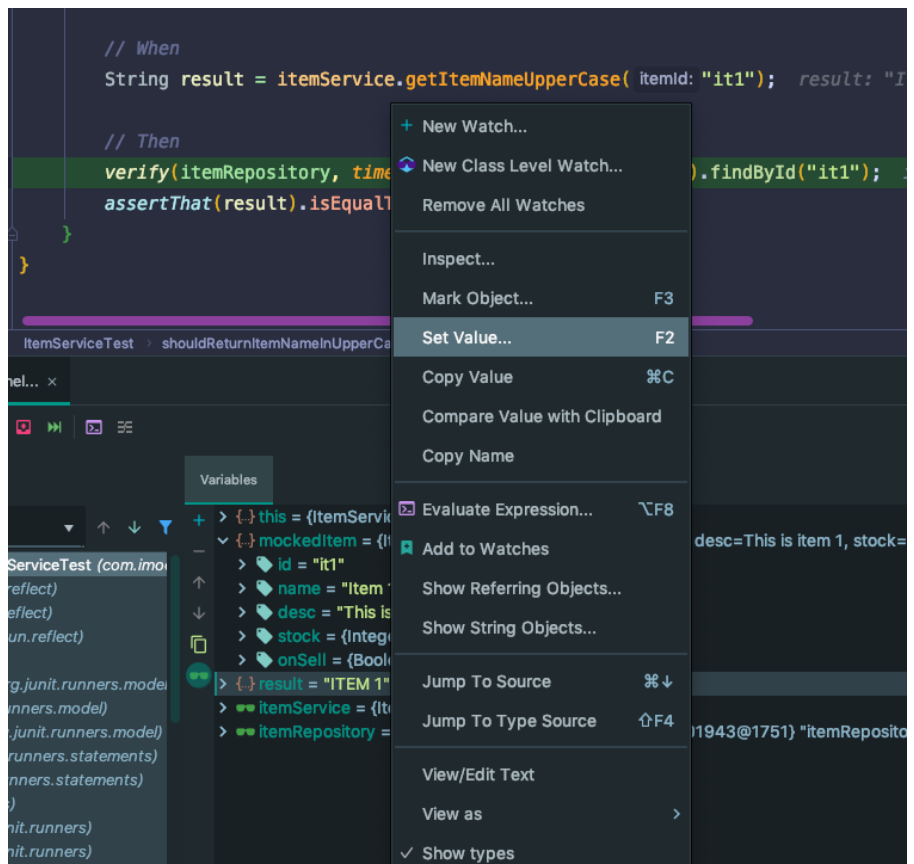
第2章 异常日志

18 一些异常处理建议

19 日志学习和使用的正确姿势



此时在 variables 选项卡中选中 `result` 变量，然后右键，选择“set value”菜单，即可对变量的值进行修改。



修改后可以继续调试观察运行结果。

4.5 表达式

目录

第1章 编码

01 开篇词：为什么学习本专栏 [已学完](#)

02 Integer缓存问题分析

03 Java序列化引发的血案

04 学习浅拷贝和深拷贝的正确方式 [已学完](#)

05 分层领域模型使用解读 [已学完](#)

06 Java属性映射的正确姿势 [已学完](#)

07 过期类、属性、接口的正确处理姿势 [已学完](#)

08 空指针引发的血案 [已学完](#)

09 当switch遇到空指针

10 枚举类的正确学习方式

11 ArrayList的subList和Arrays的asList学习

12 添加注释的正确姿势

13 你真得了解可变参数吗？

14 集合去重的正确姿势

15 学习线程池的正确姿势

16 虚拟机退出时机问题研究

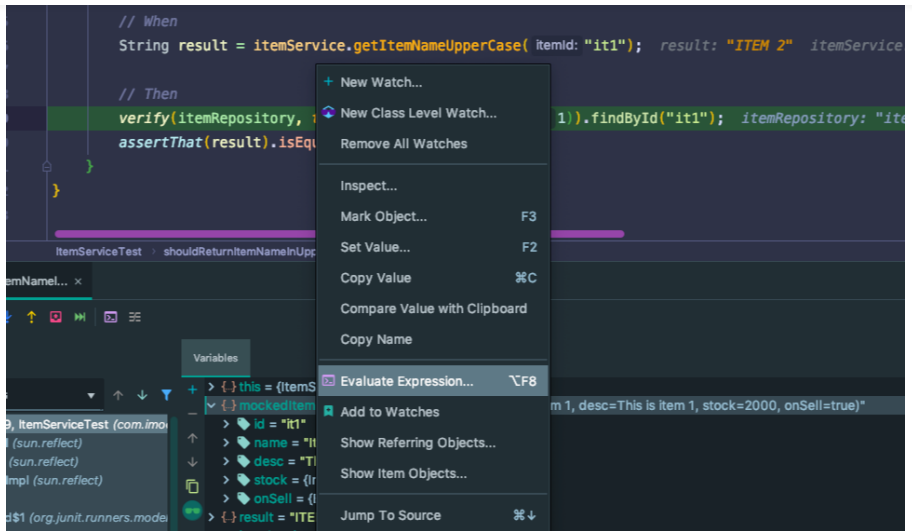
17 如何解决条件语句的多层嵌套问题？

加餐1：工欲善其事必先利其器

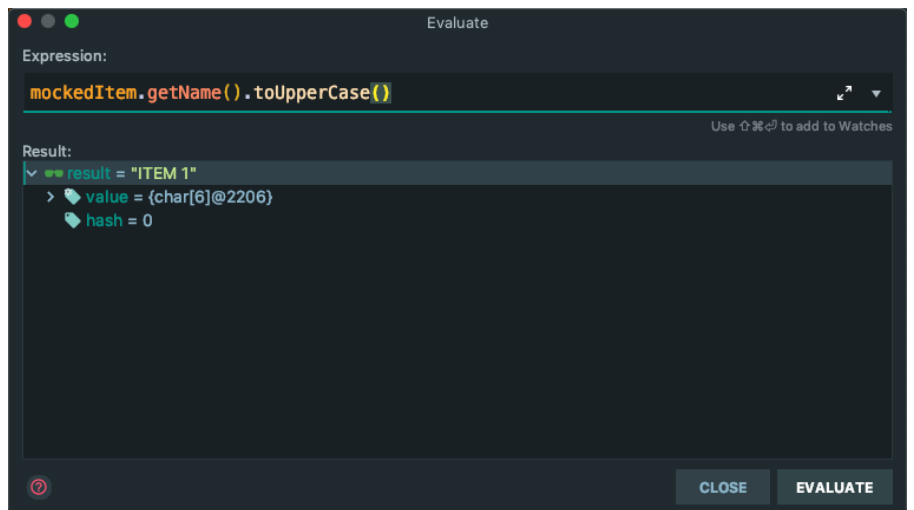
第2章 异常日志

18 一些异常处理建议

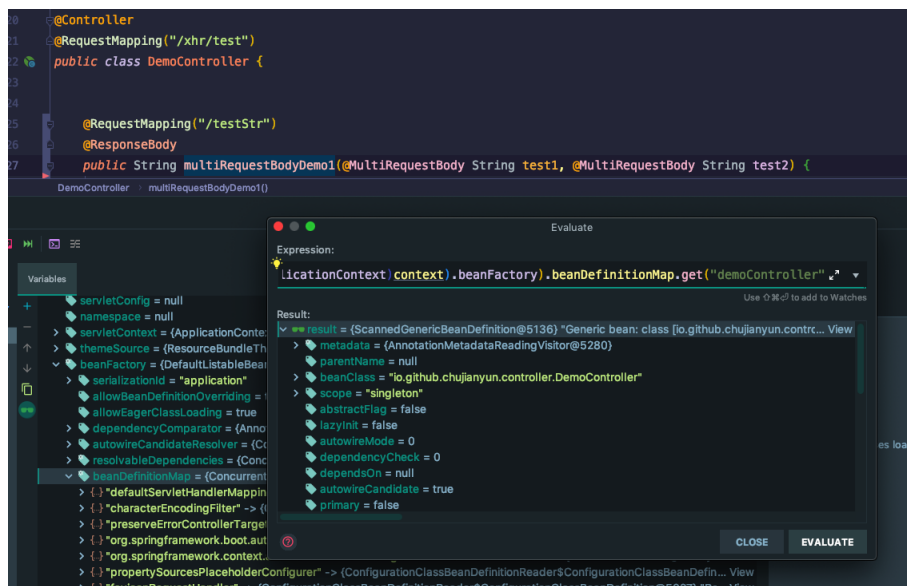
19 日志学习和使用的正确姿势



如图所示，我们可以对 `mockedItem` 变量执行表达式并查看结果：



比如我想查看 `spring` 的上下文的 `beanFactory` 中是否有名为 `demoController` 的 `bean` 定义映射，可以使用功能该功能查看：



目录

第1章 编码

01 开篇词：为什么学习本专栏 已学完

02 Integer缓存问题分析

03 Java序列化引发的血案

04 学习浅拷贝和深拷贝的正确姿势 已学完

05 分层领域模型使用解读 已学完

06 Java属性映射的正确姿势 已学完

07 过期类、属性、接口的正确处理姿势 已学完

08 空指针引发的血案 已学完

09 当switch遇到空指针

10 枚举类的正确学习方式

11 ArrayList的subList和Arrays的asList学习

12 添加注释的正确姿势

13 你真得了解可变参数吗？

14 集合去重的正确姿势

15 学习线程池的正确姿势

16 虚拟机退出时机问题研究

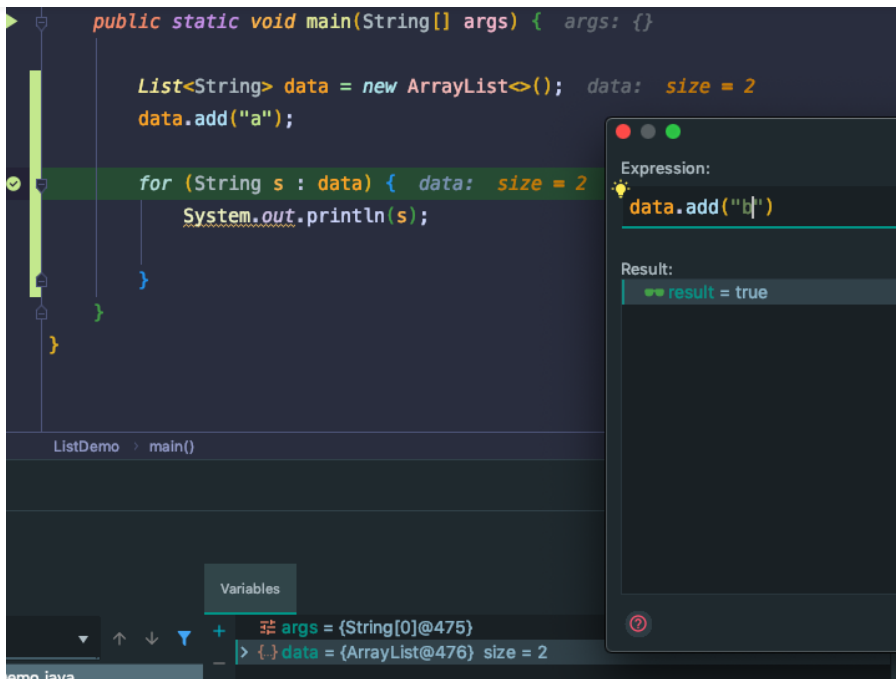
17 如何解决条件语句的多层嵌套问题？

加餐1：工欲善其事必先利其器

第2章 异常日志

18 一些异常处理建议

19 日志学习和使用的正确姿势



大家可以根据实际的情况，灵活运用。

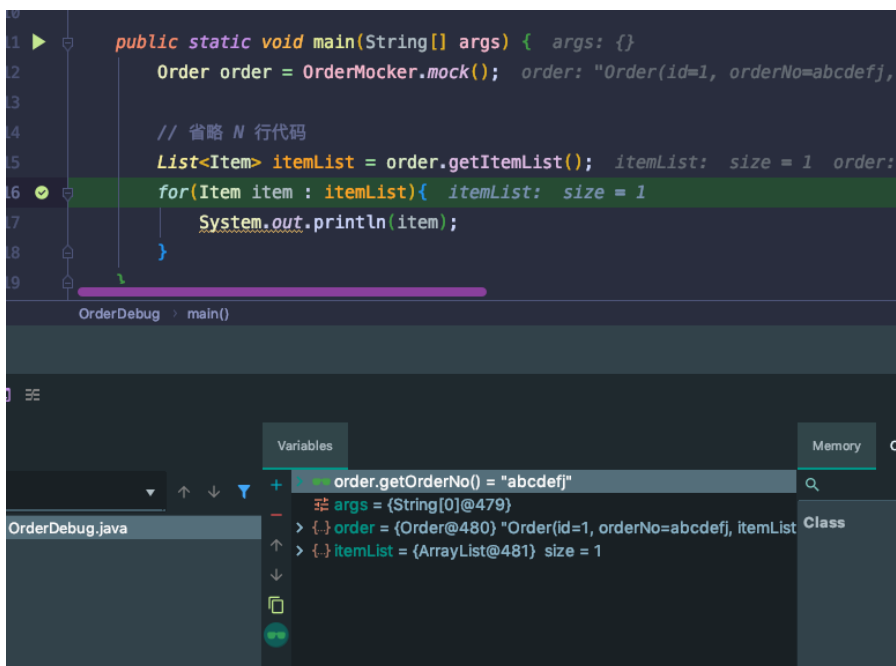
4.6 watch

如果我们在调试过程中查看某个对象的某个属性，总是使用表达式很不方便，是否可以将表达式计算的结果总是显示在变量区域呢？

答案是有的，使用 watch 功能即可实现。

在变量区右键 -> "New Watch" -> 输入想要观察的表达式即可。

如下图所示，我们可以输入 “order.getOrderNo ()”，这样就不需要调试时总展开 Order 对象来查看订单编号了。



目录

第1章 编码

01 开篇词：为什么学习本专栏 已学完

02 Integer缓存问题分析

03 Java序列化引发的血案

04 学习浅拷贝和深拷贝的正确方式 已学完

05 分层领域模型使用解读 已学完

06 Java属性映射的正确姿势 已学完

07 过期类、属性、接口的正确处理姿势 已学完

08 空指针引发的血案 已学完

09 当switch遇到空指针

10 枚举类的正确学习方式

11 ArrayList的subList和Arrays的asList学习

12 添加注释的正确姿势

13 你真得了解可变参数吗？

14 集合去重的正确姿势

15 学习线程池的正确姿势

16 虚拟机退出时机问题研究

17 如何解决条件语句的多层嵌套问题？

加餐1：工欲善其事必先利其器

第2章 异常日志

18 一些异常处理建议

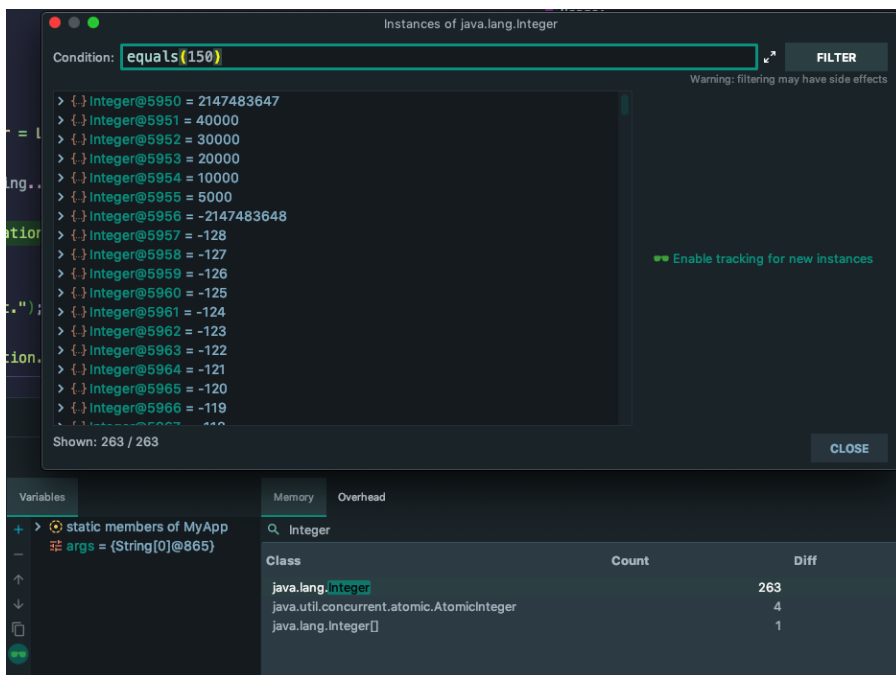
19 日志学习和使用的正确姿势

4.7 有内存对象

比如我们想通过代码调试来研究下面的示例代码共产生了几个值为 150 的对象：

```
public static void main(String[] args) {
    Integer c = 150;
    System.out.println(c==150);
}
```

我们可以在 Memory 选项栏下，搜索 Integer 就可以看到该类对象的数量，双击就可以通过表达式来过滤，非常强大。



4.8 异常断点

有些朋友可能遇到过这种问题，在一个循环中有一个数据报错，想在报错的时候断点，无法使用条件断点，而且循环次数很多，一次一次断掉放过非常麻烦。肿么办？困…

情况下面的示例代码：

```
public static void main(String[] args) {
    for (int i = 0; i < 100; i++) {
        some(i);
    }
}

private static void some(int i) {
    if (RandomUtils.nextBoolean()) {
        throw new IllegalArgumentException("错了");
    }
}
```

通常这种情况我们是知道异常的类型。

目录

第1章 编码

01 开篇词：为什么学习本专栏 已学完

02 Integer缓存问题分析

03 Java序列化引发的血案

04 学习浅拷贝和深拷贝的正确方式 已学完

05 分层领域模型使用解读 已学完

06 Java属性映射的正确姿势 已学完

07 过期类、属性、接口的正确处理姿势 已学完

08 空指针引发的血案 已学完

09 当switch遇到空指针

10 枚举类的正确学习方式

11 ArrayList的subList和Arrays的asList学习

12 添加注释的正确姿势

13 你真得了解可变参数吗？

14 集合去重的正确姿势

15 学习线程池的正确姿势

16 虚拟机退出时机问题研究

17 如何解决条件语句的多层嵌套问题？

加餐1：工欲善其事必先利其器

第2章 异常日志

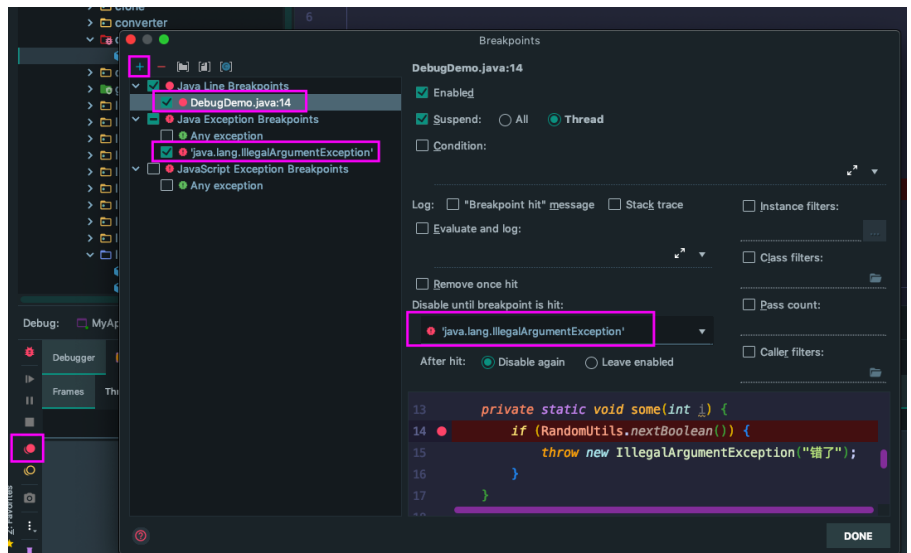
18 一些异常处理建议

19 日志学习和使用的正确姿势

第二步：我们可以点击左下角的红色断点标记，打开断点设置界面；

第三步：点击左上角的 + 号，添加 “Java Exception Breakpoints” 将 `IllegalArgumentException` 添加进去；

第四步：切换到我们的断点处，即下图所示的 `DebugDemo.java:14` 处，在 “Disable until breakpoint is hit” 处选择该异常。



此时再执行断点调试，即可捕捉到发生异常的那次调用。

同样地我们也可以通过调用栈查看整个调用过程，还可以通过移除 frame 来回退到上一层。

此外，我们还可以使用 `arthas` 的 `watch` 功能查看异常的信息。

4.9 远程调试

现在大多数公司的测试环境都会配置支持远程调试。

远程调试要求本地代码和远程服务器的代码一致，如果使用 `git`，切换到同一个分支的同一次提交即可。

设置虚拟机参数：

```
-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=8000
```

```
-Xdebug
```

在 IDEA 中 运行和调试配置中，设置 `remote` 的 `host` 和 `port` 即可。

目录

第1章 编码

01 开篇词：为什么学习本专栏 已学完

02 Integer缓存问题分析

03 Java序列化引发的血案

04 学习浅拷贝和深拷贝的正确姿势 已学完

05 分层领域模型使用解读 已学完

06 Java属性映射的正确姿势 已学完

07 过期类、属性、接口的正确处理姿势 已学完

08 空指针引发的血案 已学完

09 当switch遇到空指针

10 枚举类的正确学习方式

11 ArrayList的subList和Arrays的asList学习

12 添加注释的正确姿势

13 你真得了解可变参数吗？

14 集合去重的正确姿势

15 学习线程池的正确姿势

16 虚拟机退出时机问题研究

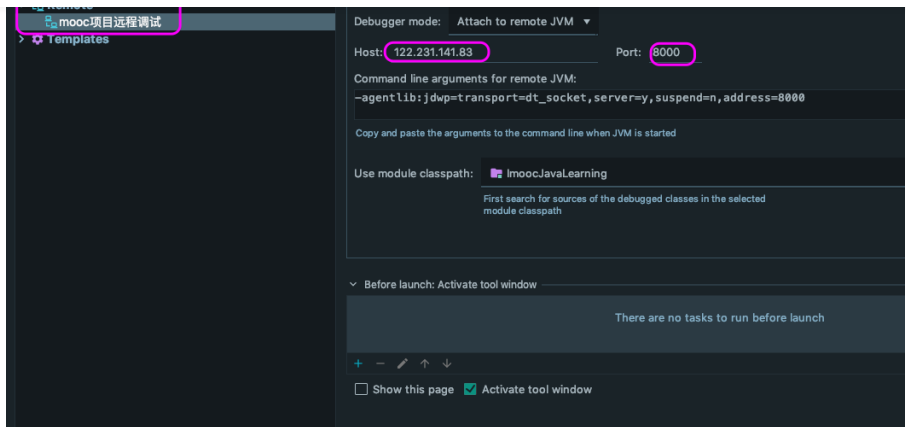
17 如何解决条件语句的多层嵌套问题？

加餐1：工欲善其事必先利其器

第2章 异常日志

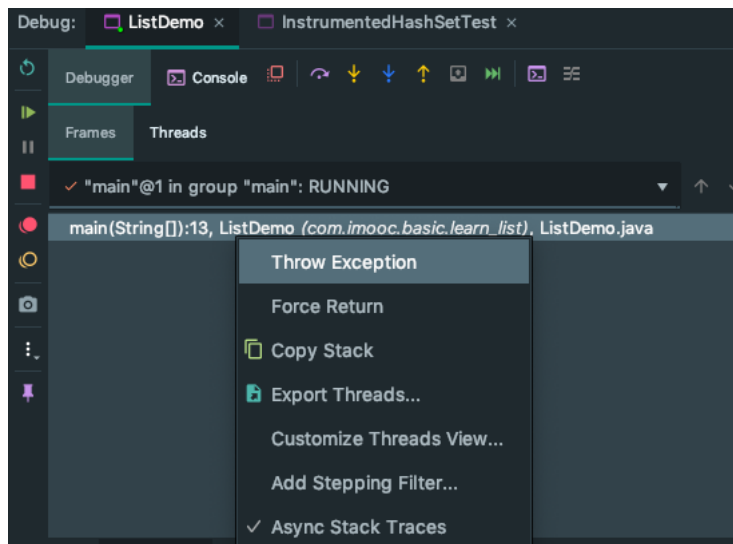
18 一些异常处理建议

19 日志学习和使用的正确姿势



4.10 其它

IDEA 的调试器非常强大，还支持调试时主动抛出异常，强制退出等功能：



希望大家在平时调试代码时，可以尝试更多新的技巧，节省时间，快速定位问题。

5. 总结

本节主要介绍了代码调试的常见用法和高级功能，掌握好调试技巧将极大提高我们排查问题的效率。

真正开发时往往是多种调试方法结合在一起，比如可以将修改变量值和回退一起使用，也可以将条件断点和修改变量值，单步等功能一起使用。

掌握好调试技巧，对快速定位问题，学习源码等都有很大的促进作用。

当然，代码调试还有很多其他的高级调试技巧可查看 IDEA 官方文档学习，也可以在开发中自行探索。

6. 课后题

课后大家编写测试代码自行练习：回退、修改变量值的功能。

参考资料

目录

第1章 编码

01 开篇词：为什么学习本专栏 [已学完](#)

02 Integer缓存问题分析

03 Java序列化引发的血案

04 学习浅拷贝和深拷贝的正确姿势 [已学完](#)

05 分层领域模型使用解读 [已学完](#)

06 Java属性映射的正确姿势 [已学完](#)

07 过期类、属性、接口的正确处理姿势 [已学完](#)

08 空指针引发的血案 [已学完](#)

09 当switch遇到空指针

10 枚举类的正确学习方式

11 ArrayList的subList和Arrays的asList学习

12 添加注释的正确姿势

13 你真得了解可变参数吗？

14 集合去重的正确姿势

15 学习线程池的正确姿势

16 虚拟机退出时机问题研究

17 如何解决条件语句的多层嵌套问题？

加餐1：工欲善其事必先利其器

第2章 异常日志

18 一些异常处理建议

19 日志学习和使用的正确姿势

← 23 Java学习宝典

25 阅读源码的正确姿势 →

精选留言 2

欢迎在这里发表留言，作者筛选后可公开显示

慕莱坞2595503

多谢老师,学习到了!

👍 1 回复

3天前

明明如月 回复 慕莱坞2595503
关键是平时开发灵活运用

回复

1天前

慕粉3543028

赞，大写的66666

👍 2 回复

5天前

明明如月 回复 慕粉3543028
多谢支持，希望学习和工作中多运用，熟练掌握

回复

4天前

千学不如一看，千看不如一练