

37 | 实战二（下）：重构ID生成器项目中各函数的异常处理代码

2020-01-27 王争

设计模式之美

[进入课程 >](#)



讲述：冯永吉

时长 10:49 大小 9.92M



平时进行软件设计开发的时候，我们除了要保证正常情况下的逻辑运行正确之外，还需要编写大量额外的代码，来处理有可能出现的异常情况，以保证代码在任何情况下，都在我们的掌控之内，不会出现非预期的运行结果。程序的 bug 往往都出现在一些边界条件和异常情况下，所以说，异常处理得好坏直接影响了代码的健壮性。全面、合理地处理各种异常能有效减少代码 bug，也是保证代码质量的一个重要手段。

在上一节课中，我们讲解了几种异常情况的处理方式，比如返回错误码、NULL 值、空对象、异常对象。针对最常用的异常对象，我们还重点讲解了两种异常类型的应用场景，☆ 针对函数抛出的异常的三种处理方式：直接吞掉、原封不动地抛出和包裹成新的异常抛出。

除此之外，在上一节课的开头，我们还针对 ID 生成器的代码，提出了 4 个有关异常处理的问题。今天，我们就用一节课的时间，结合上一节课讲到的理论知识，来逐一解答一下这几个问题。

话不多说，让我们正式开始今天的内容吧！

重构 generate() 函数

首先，我们来看，对于 generate() 函数，如果本机名获取失败，函数返回什么？这样的返回值是否合理？


 复制代码

```
1 public String generate() {
2     String substrOfHostName = getLastFiledOfHostName();
3     long currentTimeMillis = System.currentTimeMillis();
4     String randomString = generateRandomAlphameric(8);
5     String id = String.format("%s-%d-%s",
6         substrOfHostName, currentTimeMillis, randomString);
7     return id;
8 }
```

ID 由三部分构成：本机名、时间戳和随机数。时间戳和随机数的生成函数不会出错，唯独主机名有可能获取失败。在目前的代码实现中，如果主机名获取失败，substrOfHostName 为 NULL，那 generate() 函数会返回类似 “null-16723733647-83Ab3uK6” 这样的数据。如果主机名获取失败，substrOfHostName 为空字符串，那 generate() 函数会返回类似 “-16723733647-83Ab3uK6” 这样的数据。

在异常情况下，返回上面两种特殊的 ID 数据格式，这样的做法是否合理呢？这个其实很难讲，我们要看具体的业务是怎么设计的。不过，我更倾向于明确地将异常告知调用者。所以，这里最好是抛出受检异常，而非特殊值。

按照这个设计思路，我们对 generate() 函数进行重构。重构之后的代码如下所示：


 复制代码

```
1 public String generate() throws IdGenerationFailureException {
2     String substrOfHostName = getLastFiledOfHostName();
3     if (substrOfHostName == null || substrOfHostName.isEmpty()) {
4         throw new IdGenerationFailureException("host name is empty.");
5     }
6 }
```

```
5     }
6     long currentTimeMillis = System.currentTimeMillis();
7     String randomString = generateRandomAlphameric(8);
8     String id = String.format("%s-%d-%s",
9         substrOfHostName, currentTimeMillis, randomString);
10    return id;
11 }
```

重构 getLastFiledOfHostName() 函数

对于 getLastFiledOfHostName() 函数，是否应该将 UnknownHostException 异常在函数内部吞掉（try-catch 并打印日志），还是应该将异常继续往上抛出？如果往上抛出的话，是直接把 UnknownHostException 异常原封不动地抛出，还是封装成新的异常抛出？

 复制代码

```
1 private String getLastFiledOfHostName() {
2     String substrOfHostName = null;
3     try {
4         String hostName = InetAddress.getLocalHost().getHostName();
5         substrOfHostName = getLastSubstrSplittedByDot(hostName);
6     } catch (UnknownHostException e) {
7         logger.warn("Failed to get the host name.", e);
8     }
9     return substrOfHostName;
10 }
```

现在的处理方式是当主机名获取失败的时候，getLastFiledOfHostName() 函数返回 NULL 值。我们前面讲过，是返回 NULL 值还是异常对象，要看获取不到数据是正常行为，还是异常行为。获取主机名失败会影响后续逻辑的处理，并不是我们期望的，所以，它是一种异常行为。这里最好是抛出异常，而非返回 NULL 值。

至于是直接将 UnknownHostException 抛出，还是重新封装成新的异常抛出，要看函数跟异常是否有业务相关性。getLastFiledOfHostName() 函数用来获取主机名的最后一个字段，UnknownHostException 异常表示主机名获取失败，两者算是业务相关，所以可以直接将 UnknownHostException 抛出，不需要重新包裹成新的异常。

按照上面的设计思路，我们对 getLastFiledOfHostName() 函数进行重构。重构后的代码如下所示：

[复制代码](#)

```
1 private String getLastFiledOfHostName() throws UnknownHostException{
2     String substrOfHostName = null;
3     String hostName = InetAddress.getLocalHost().getHostName();
4     substrOfHostName = getLastSubstrSplittedByDot(hostName);
5     return substrOfHostName;
6 }
```

getLastFiledOfHostName() 函数修改之后，generate() 函数也要做相应的修改。我们需要在 generate() 函数中，捕获 getLastFiledOfHostName() 抛出的 UnknownHostException 异常。当我们捕获到这个异常之后，应该怎么办呢？

按照之前的分析，ID 生成失败的时候，我们需要明确地告知调用者。所以，我们不能在 generate() 函数中，将 UnknownHostException 这个异常吞掉。那我们应该原封不动地抛出，还是封装成新的异常抛出呢？

我们选择后者。在 generate() 函数中，我们需要捕获 UnknownHostException 异常，并重新包裹成新的异常 IdGenerationFailureException 往上抛出。之所以这么做，有下面三个原因。

调用者在使用 generate() 函数的时候，只需要知道它生成的是随机唯一 ID，并不关心 ID 是如何生成的。也就是说，这是依赖抽象而非实现编程。如果 generate() 函数直接抛出 UnknownHostException 异常，实际上是暴露了实现细节。

从代码封装的角度来讲，我们不希望将 UnknownHostException 这个比较底层的异常，暴露给更上层的代码，也就是调用 generate() 函数的代码。而且，调用者拿到这个异常的时候，并不能理解这个异常到底代表了什么，也不知道该如何处理。

UnknownHostException 异常跟 generate() 函数，在业务概念上没有相关性。

按照上面的设计思路，我们对 generate() 的函数再次进行重构。重构后的代码如下所示：

[复制代码](#)

```
1 public String generate() throws IdGenerationFailureException {
2     String substrOfHostName = null;
3     try {
4         substrOfHostName = getLastFiledOfHostName();
5     } catch (UnknownHostException e) {
6         throw new IdGenerationFailureException("host name is empty.");
7     }
8 }
```

```
7     }
8     long currentTimeMillis = System.currentTimeMillis();
9     String randomString = generateRandomAlphameric(8);
10    String id = String.format("%s-%d-%s",
11        substrOfHostName, currentTimeMillis, randomString);
12    return id;
13 }
```

重构 getLastSubstrSplittedByDot() 函数

对于 getLastSubstrSplittedByDot(String hostName) 函数，如果 hostName 为 NULL 或者空字符串，这个函数应该返回什么？

 复制代码

```
1  @VisibleForTesting
2  protected String getLastSubstrSplittedByDot(String hostName) {
3      String[] tokens = hostName.split("\\.");
4      String substrOfHostName = tokens[tokens.length - 1];
5      return substrOfHostName;
6  }
```


理论上讲，参数传递的正确性应该有程序员来保证，我们无需做 NULL 值或者空字符串的判断和特殊处理。调用者本不应该把 NULL 值或者空字符串传递给 getLastSubstrSplittedByDot() 函数。如果传递了，那就是 code bug，需要修复。但是，话说回来，谁也保证不了程序员就一定不会传递 NULL 值或者空字符串。那我们到底该不该做 NULL 值或空字符串的判断呢？

如果函数是 private 类私有的，只在类内部被调用，完全在你自己的掌控之下，自己保证在调用这个 private 函数的时候，不要传递 NULL 值或空字符串就可以了。所以，我们可以不在 private 函数中做 NULL 值或空字符串的判断。如果函数是 public 的，你无法掌控会被谁调用以及如何调用（有可能某个同事一时疏忽，传递进了 NULL 值，这种情况也是存在的），为了尽可能提高代码的健壮性，我们最好是在 public 函数中做 NULL 值或空字符串的判断。

那你可能会说，getLastSubstrSplittedByDot() 是 protected 的，既不是 private 函数，也不是 public 函数，那要不要做 NULL 值或空字符串的判断呢？

之所以将它设置为 `protected`，是为了方便写单元测试。不过，单元测试可能要测试一些 corner case，比如输入是 `NULL` 值或者空字符串的情况。所以，这里我们最好也加上 `NULL` 值或空字符串的判断逻辑。虽然加上有些冗余，但多加些检验总归不会错的。

按照这个设计思路，我们对 `getLastSubstrSplittedByDot()` 函数进行重构。重构之后的代码如下所示：

 复制代码

```
1  @VisibleForTesting
2  protected String getLastSubstrSplittedByDot(String hostName) {
3      if (hostName == null || hostName.isEmpty()) {
4          throw IllegalArgumentException("..."); //运行时异常
5      }
6      String[] tokens = hostName.split("\\.");
7      String substrOfHostName = tokens[tokens.length - 1];
8      return substrOfHostName;
9  }
```

按照上面讲的，我们在使用这个函数的时候，自己也要保证不传递 `NULL` 值或者空字符串进去。所以，`getLastFiledOfHostName()` 函数的代码也要作相应的修改。修改之后的代码如下所示：

 复制代码

```
1  private String getLastFiledOfHostName() throws UnknownHostException{
2      String substrOfHostName = null;
3      String hostName = InetAddress.getLocalHost().getHostName();
4      if (hostName == null || hostName.isEmpty()) { // 此处做判断
5          throw new UnknownHostException("...");
6      }
7      substrOfHostName = getLastSubstrSplittedByDot(hostName);
8      return substrOfHostName;
9  }
```

重构 `generateRandomAlphameric()` 函数

对于 `generateRandomAlphameric(int length)` 函数，如果 `length < 0` 或 `length = 0`，这个函数应该返回什么？

```

1  @VisibleForTesting
2  protected String generateRandomAlphameric(int length) {
3      char[] randomChars = new char[length];
4      int count = 0;
5      Random random = new Random();
6      while (count < length) {
7          int maxAscii = 'z';
8          int randomAscii = random.nextInt(maxAscii);
9          boolean isDigit= randomAscii >= '0' && randomAscii <= '9';
10         boolean isUppercase= randomAscii >= 'A' && randomAscii <= 'Z';
11         boolean isLowercase= randomAscii >= 'a' && randomAscii <= 'z';
12         if (isDigit|| isUppercase || isLowercase) {
13             randomChars[count] = (char) (randomAscii);
14             ++count;
15         }
16     }
17     return new String(randomChars);
18 }
19 }

```

我们先来看 `length < 0` 的情况。生成一个长度为负值的随机字符串是不符合常规逻辑的，是一种异常行为。所以，当传入的参数 `length < 0` 的时候，我们抛出 `IllegalArgumentException` 异常。

我们再来看 `length = 0` 的情况。`length = 0` 是否是异常行为呢？这就看你自己怎么定义了。我们既可以把它定义为一种异常行为，抛出 `IllegalArgumentException` 异常，也可以把它定义为一种正常行为，让函数在入参 `length = 0` 的情况下，直接返回空字符串。不管选择哪种处理方式，最关键的一点是，要在函数注释中，明确告知 `length = 0` 的情况下，会返回什么样的数据。

重构之后的 RandomIdGenerator 代码

对 `RandomIdGenerator` 类中各个函数异常情况处理代码的重构，到此就结束了。为了方便查看，我把重构之后的代码，重新整理之后贴在这里了。你可以对比着看一下，跟你的重构思路是否一致。

```

1  public class RandomIdGenerator implements IdGenerator {
2      private static final Logger logger = LoggerFactory.getLogger(RandomIdGenerat
3
4      @Override

```

```

5  public String generate() throws IdGenerationFailureException {
6      String substrOfHostName = null;
7      try {
8          substrOfHostName = getLastFiledOfHostName();
9      } catch (UnknownHostException e) {
10         throw new IdGenerationFailureException("...", e);
11     }
12     long currentTimeMillis = System.currentTimeMillis();
13     String randomString = generateRandomAlphameric(8);
14     String id = String.format("%s-%d-%s",
15         substrOfHostName, currentTimeMillis, randomString);
16     return id;
17 }
18
19 private String getLastFiledOfHostName() throws UnknownHostException{
20     String substrOfHostName = null;
21     String hostName = InetAddress.getLocalHost().getHostName();
22     if (hostName == null || hostName.isEmpty()) {
23         throw new UnknownHostException("...");
24     }
25     substrOfHostName = getLastSubstrSplittedByDot(hostName);
26     return substrOfHostName;
27 }
28
29 @VisibleForTesting
30 protected String getLastSubstrSplittedByDot(String hostName) {
31     if (hostName == null || hostName.isEmpty()) {
32         throw new IllegalArgumentException("...");
33     }
34
35     String[] tokens = hostName.split("\\.");
36     String substrOfHostName = tokens[tokens.length - 1];
37     return substrOfHostName;
38 }
39
40 @VisibleForTesting
41 protected String generateRandomAlphameric(int length) {
42     if (length <= 0) {
43         throw new IllegalArgumentException("...");
44     }
45
46     char[] randomChars = new char[length];
47     int count = 0;
48     Random random = new Random();
49     while (count < length) {
50         int maxAscii = 'z';
51         int randomAscii = random.nextInt(maxAscii);
52         boolean isDigit= randomAscii >= '0' && randomAscii <= '9';
53         boolean isUppercase= randomAscii >= 'A' && randomAscii <= 'Z';
54         boolean isLowercase= randomAscii >= 'a' && randomAscii <= 'z';
55         if (isDigit|| isUppercase || isLowercase) {
56             randomChars[count] = (char) (randomAscii);

```



```
57         ++count;
58     }
59 }
60     return new String(randomChars);
61 }
62 }
```

重点回顾

好了，今天的内容到此就讲完了。我们一块来总结回顾一下，你需要重点掌握的内容。

今天的内容比较偏实战，是对上节课学到的理论知识的一个应用。从今天的实战中，你学到了哪些更高层的软件设计和开发思想呢？我这里抛砖引玉，总结了下面 3 点。

再简单的代码，看上去再完美的代码，只要我们下功夫去推敲，总有可以优化的空间，就看你愿不愿把事情做到极致。

如果你内功不够深厚，理论知识不够扎实，那你就很难参透开源项目的代码到底优秀在哪里。就像如果我们没有之前的理论学习，没有今天我给你一点一点重构、讲解、分析，只是给你最后重构好的 RandomIdGenerator 的代码，你真的能学到它的设计精髓吗？

对比 [🔗第 34 节课](#)最初小王的 IdGenerator 代码和最终的 RandomIdGenerator 代码，它们一个是“能用”，一个是“好用”，天壤之别。作为一名程序员，起码对代码要有追求啊，不然跟咸鱼有啥区别！

课堂讨论

我们花了 4 节课的时间，对一个非常简单的、不到 40 行的 ID 生成器代码，做了多次迭代重构。除了刚刚我在“重点回顾”中讲到的那几点之外，从这个迭代重构的过程中，你还学到哪些更有价值的东西？

欢迎在留言区写下你的思考和想法，和同学一起交流和分享。如果有收获，也欢迎你把这篇文章分享给你的朋友。

点击参加小程序学习打卡 

8个月，攻克设计模式



扫一扫参与小程序打卡



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 36 | 实战二（上）：程序出错该返回啥？NULL、异常、错误码、空对象？

下一篇 38 | 总结回顾面向对象、设计原则、编程规范、重构技巧等知识点

精选留言 (19)

 写留言



Jxin

2020-01-27

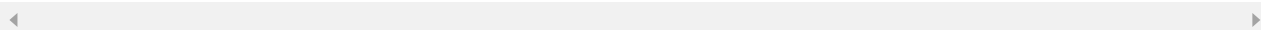
还学到什么：

1.一下子想搞个例子讲这些真的太难了，拍着脑子想demo。栏主这个demo背景简单，也将要讲的内容串起来了，实属不易，辛苦栏主了。

个人见解：...

展开 

作者回复：我觉得你懂我~



11



皮卡皮卡

2020-01-28

争哥这种设计思路考虑了一下，但是在业务中往往获取唯一ID的地方，不关心ID内部生成错误，需要的只是能够返回出来ID即可。目前的处理是异常在generate内部自己解决，同时返回ID

1

3



your problem?

2020-01-27

打卡，也祝大家新年快乐，身体健康，另外我始终觉得generateRandomAlphameric这个函数里，随机获取这个写法很不利于性能测试，假如这个函数会被百万，甚至千万次的调用，不可控性也太强了，我觉得可以改成随机生成0-26的数字，对应去加到字母的位置，不知道老师和大家有什么想法吗

展开

2

2



undefined

2020-02-01

个人见解：如果 id 生成器需要应用到生产环境，类似 hostname 获取失败的问题，需要由生成器本身给出降级方案。一来为了 id 格式统一，二来假若抛给业务，业务对于这种系统底层的失败，也没有什么好的解决方法。

展开

1

1



Yang

2020-01-28

还学到了：

1.函数出错时是返回NULL还是异常对象？

要看获取不到数据是正常行为，还是异常行为，如果业务上来说是异常行为就抛出异常，反之返回NULL。

...

展开

1

1



Frank

2020-01-27

今天学习了异常代码处理思路。在处理到异常时，通常会将上层关心的异常直接包装成RuntimeException往上抛，没有根据业务域定义相关的自定义异常。通过今天的学习，了解到处理异常的基本思路：是往上抛还是吞掉，主要看调用者是否关心该异常。是否要包装成新的异常主要看调用者是否理解该异常，该异常是否业务相关。如果能理解、业务相关可以直接抛，否则重新包装。...

展开 ∨



1



高源

2020-01-27

希望老师每节课举的代码有下载的地方，自己下载下来结合老师讲解的，自己理解体会其中的解决问题

作者回复: 好的，等我俩月，我整理好，一块放到github上：

<https://github.com/wangzheng0822>



1



Geek_kobe

2020-01-27

果然还是看技术文章能让恐慌的心静下来

展开 ∨



1



Demon.Lee

2020-02-02

小伙伴们，针对函数入参的值，里面可能含有前后空格，你们会检验么，好纠结🤔



batman

2020-02-01

以前不管啥情况就抛出个Exception搞得很懵逼！

函数和异常的业务相关性分析得很透彻，赞！

展开 ∨



守拙

2020-01-30

很喜欢作者对于"正常情况"和"异常情况"判定的讲解，还有依赖抽象而不是具体的思想在以上直接上抛还是包装后上抛的分别。

展开 ∨





黄林晴

2020-01-28

打卡

展开 ▾



liu_liu

2020-01-28

写代码不是糊弄，写出好的有水平的代码需要下一番功夫。对代码保持敬畏之心，有追求极致的思想，才会越来越好。

展开 ▾



javaadu

2020-01-27

我再提一点自己的改进想法：修改后的代码里，generate方法还应该处理掉8这个魔法数字，如果需要根据用户定制长度，则需要提供另一个不带默认值的方法，并在generate方法里处理随机方法抛出的参数非法异常



辣么大

2020-01-27

学到什么：

- 1、动手实践。争哥的这4节看着简单，实际信息量很大，动手实践一下能有更深的体会。
- 2、跑去研究了Java Random的源码的构造函数如何实现的。
搞清random做为变量可以放在哪里使用，以及random.nextInt()的取值范围。
- 3、异常如何处理：什么时候吞掉、抛出去，或者包装后抛出去。...

展开 ▾



ちょくん

2020-01-27

滴滴打卡

展开 ▾



小晏子

2020-01-27

感觉generate函数失败后报错信息是host name empty会有点奇怪，意味着用户还要去设置服务器的hostname，程序和服务器设置有了依赖性了，如果用户不会设置或者是基于容

器的，那还要用户花费时间去设置主机名，感觉不好。

展开 ▾



Harvey

2020-01-27

设计之所以难是因为没有标准答案，很多权衡是依赖于具体业务的。这就是DDD的思想所在，要先想清楚问题域是什么在思考解决方案。很多开发讨论问题的时候没有层次，上来就陷入技术细节，这就叫缺乏抽象。下游系统要想清楚哪些是上游系统给你提供的服务？哪些是人家的内部技术实现？比如ID生成，作为上游系统，ID生成服务提供的是有小概率重复的随机ID服务，至于随机算法，下游系统不必关心，这是上游系统的内部实现，这...

展开 ▾



Jeff.Smile

2020-01-27

大年初三，抢占沙发！

展开 ▾

