

# 令人匪夷所思的 Magic 之魔数,你真的不会

原创：子悠 Java极客技术 5天前

点击蓝色“[Java极客技术](#)”关注我哟  
加个“[星标](#)”，一起快乐成长



## 01、故事背景

试想一下你现在有个业务场景需要你识别出各种类型的文件，然后进行不同的处理，这些文件不管是用户上传还是怎么来的，你都需要知道文件是什么类型，比如是一张 **JPG** 图片，还是一个 **GIF** 图片又或者是 **PDF** 等等其他类型的文件。这个时候你会想，这还不简单么，几行 **Java** 代码就搞定了。

如下：

```
1      public static String getFileTypeByExt(String filePath) {  
2          if (StringUtils.isBlank(filePath)) {  
3              return null;  
4          }  
5          return filePath.toLowerCase().substring(filePath.lastIndexOf(".") + 1);  
6      }
```

敲完过后，满心欢喜，这个需求简单的无法用言语去描述，提交了代码，更新到测试环境去了，优雅又不是风度的等着下班。

眼看马上到下班的点了，起身收拾东西准备下班，正在你关上电脑的那一刹那，说时迟那时快，测试人员突然跑过来说有个 bug，需要你看下。此时的你脑海中浮现了如下场景：

1. bug? 怎么可能，我写的代码怎么可能有 bug?
2. 什么 bug? 肯定是你不会用!
3. 肯定没清缓存吧（但是我改的后台代码，好像不影响浏览器缓存啊，摸了下后脑勺~）
4. 不会真有 bug 吧~~

几种情况快速的在你大脑中闪现过，但是你很自信的认为是测试人员不会用，不过你没有表现出来，淡定并且很有礼貌的说了句：哦，是么，我跟你过去看下。

走到测试人员的工位前，看着桌上有一个 jpg 图片，点击按钮上传，选中这个图片，开始上传，一顿操作猛如虎，正在准备大功告成的时候，系统弹出了一个“系统错误”的弹框出来，登上测试服务器看了日志信息，提示在处理图片的时候相关类库提示失败了。

咦，不科学啊，不会是真的，清下缓存，刷新下页面，循环重复了几次操作之后，结果还是一样的，这个时候你有点认为这是个 bug 了。但是自己回想了一下差不多半个小时前写的代码，怎么都不会有问题啊，要不换个图片试试？

就这样抱着试一试的想法，换了一张图片，惊喜的发现成功了，没有提示系统错误，正常的上传成功了。看到这个现象，内心欢喜了一下，原来是图片的问题，就说不是 bug 了。

但是等等，为什么换个图片可以，这个图片就不行呢？仔细看了一个原来的图片文件，没发现什么异常啊，就是图片的大小差不多 5M，换的图片是随便截图的只有几百 k，难道是大小的原因？但是不对啊，系统是支持上传 PDF 文件的，所以大小是可以超过 5M 的啊。

咦，等等这个文件的大小怎么跟我刚刚发的 PDF 的大小一样啊，之前测试文档上传，发了一个 PDF 给测试人员，大小差不多就是这个，怎么也有一个这样大小的图片呢？然后随口问了句：你这个图片哪来的啊？只见测试人员淡定的说到：哦，我是把刚刚的 PDF 文件的后缀改成了 JPG，所以这个图片不是真的图片，我只是改了一下后缀。

听到这句话的时候你的内心是崩溃的，但是你没有表现出来，毕竟测试人员的职责就是维护系统的稳定性。

故事发生到这里，你已经知道不能使用上面那段优雅的代码了，需要考虑其他方案，简单的根据后缀名来判断文件类型进行处理是不行了了。

此时的你陷入了深深的沉思当中。。。

在走向工位的路上拿出手机给媳妇（假如有的话）发了个微信：今晚加班，不回去吃饭了。

## 02、解决方案

为了解决这个问题，在面向搜索引擎编程的年代，你在网上搜到了一个叫做魔数的东西（本文的重点来了），抱着试一试的想法，实践起来了。

### 2.1、魔数的定义

魔数有很多种定义，这里我们讨论的主要是在编程领域的定义

文件的起始几个字节的内容是固定的（或是有意填充，或是本就如此），这几个字节的内容也被称为**魔数**（magic number），因此可以根据这几个字节的内容确定文件类型。

### 2.2、常见文件类型的魔数

```
1 package com.coocaa.ad.caeser.engine.com.test.model;
2
3 /**
4  * <br>
5  * <b>Function: </b><br>
6  * <b>Author: </b>@author ziyou<br>
7  * <b>Date: </b>2019-10-24 11:21<br>
8  * <b>Desc: </b>无<br>
9  */
10 public enum FileType {
11     /**
12      * JPEG
13      */
14     JPEG("JPEG", "FFD8FF"),
15
16     /**
17      * PNG
```

```
18     */
19     PNG("PNG", "89504E47"),
20
21     /**
22     * GIF
23     */
24     GIF("GIF", "47494638"),
25
26     /**
27     * TIFF
28     */
29     TIFF("TIFF", "49492A00"),
30
31     /**
32     * Windows bitmap
33     */
34     BMP("BMP", "424D"),
35
36     /**
37     * CAD
38     */
39     DWG("DWG", "41433130"),
40
41     /**
42     * Adobe photoshop
43     */
44     PSD("PSD", "38425053"),
45
46     /**
47     * Rich Text Format
48     */
49     RTF("RTF", "7B5C727466"),
50
51     /**
52     * XML
53     */
54     XML("XML", "3C3F786D6C"),
55
56     /**
57     * HTML
```

```
58      */
59      HTML("HTML", "68746D6C3E"),
60
61      /**
62      * Outlook Express
63      */
64      DBX("DBX", "CFAD12FEC5FD746F "),
65
66      /**
67      * Outlook
68      */
69      PST("PST", "2142444E"),
70
71      /**
72      * doc;xls;dot;ppt;xla;ppa;pps;pot;msi;sdw;db
73      */
74      OLE2("OLE2", "0xD0CF11E0A1B11AE1"),
75
76      /**
77      * Microsoft Word/Excel
78      */
79      XLS_DOC("XLS_DOC", "D0CF11E0"),
80
81      /**
82      * Microsoft Access
83      */
84      MDB("MDB", "5374616E64617264204A"),
85
86      /**
87      * Word Perfect
88      */
89      WPB("WPB", "FF575043"),
90
91      /**
92      * Postscript
93      */
94      EPS_PS("EPS_PS", "252150532D41646F6265"),
95
96      /**
97      * Adobe Acrobat
```

```
98      */
99      PDF("PDF", "255044462D312E"),
100
101     /**
102      * Windows Password
103      */
104     PWL("PWL", "E3828596"),
105
106     /**
107      * ZIP Archive
108      */
109     ZIP("ZIP", "504B0304"),
110
111     /**
112      * ARAR Archive
113      */
114     RAR("RAR", "52617221"),
115
116     /**
117      * WAVE
118      */
119     WAV("WAV", "57415645"),
120
121     /**
122      * AVI
123      */
124     AVI("AVI", "41564920"),
125
126     /**
127      * Real Audio
128      */
129     RAM("RAM", "2E7261FD"),
130
131     /**
132      * Real Media
133      */
134     RM("RM", "2E524D46"),
135
136     /**
137      * Quicktime
```

```
138     */
139     MOV("MOV", "6D6F6F76"),
140
141     /**
142     * Windows Media
143     */
144     ASF("ASF", "3026B2758E66CF11"),
145
146     /**
147     * MIDI
148     */
149     MID("MID", "4D546864");
150
151     private String key;
152     private String value;
153
154     FileType(String key, String value) {
155         this.key = key;
156         this.value = value;
157     }
158
159     public String getValue() {
160         return value;
161     }
162     public String getKey() {
163         return key;
164     }
165 }
166
```

## 2.3、工具类

```
1 package com.test.util;
2
3 import java.io.FileInputStream;
4 import java.io.IOException;
5 import java.io.InputStream;
6
```

```
7  /**
8   * <br>
9   * <b>Function: </b><br>
10  * <b>Author: </b>@author ziyou<br>
11  * <b>Date: </b>2019-10-24 11:16<br>
12  * <b>Desc: </b>无<br>
13  */
14  public class FileUtil {
15
16      /**
17       * 获取文件投
18       *
19       * @param filePath 文件路径
20       * @return 16 进制的文件投信息
21       * @throws IOException
22       */
23      private static String getFileHeader(String filePath) throws IOException {
24          byte[] b = new byte[28];
25          InputStream inputStream = new FileInputStream(filePath);
26          inputStream.read(b, 0, 28);
27          inputStream.close();
28          return bytes2hex(b);
29      }
30
31      /**
32       * 将字节数组转换成16进制字符串
33       */
34      private static String bytes2hex(byte[] src) {
35          StringBuilder stringBuilder = new StringBuilder("");
36          if (src == null || src.length <= 0) {
37              return null;
38          }
39          for (byte b : src) {
40              int v = b & 0xFF;
41              String hv = Integer.toHexString(v);
42              if (hv.length() < 2) {
43                  stringBuilder.append(0);
44              }
45              stringBuilder.append(hv);
46          }
```



```
47         return stringBuilder.toString();
48     }
49
50     /**
51     * 根据文件路径获取文件类型
52     *
53     * @param filePath 文件路径
54     * @return 文件类型
55     * @throws IOException
56     */
57     public static FileType getFileType(String filePath) throws IOException {
58         String fileHead = getFileHeader(filePath);
59         if (null == fileHead || fileHead.length() == 0) {
60             return null;
61         }
62         fileHead = fileHead.toUpperCase();
63         FileType[] fileTypes = FileType.values();
64         for (FileType type : fileTypes) {
65             if (fileHead.startsWith(type.getValue())) {
66                 return type;
67             }
68         }
69         return null;
70     }
71 }
72
```

## 2.4、使用

```
1     public static void main(String[] args) throws IOException {
2         //         String filePath = "/Users/ziyou/Downloads/SpringBoot实战.pdf";
3         String filePath = "/Users/ziyou/Downloads/SpringBoot实战.png";
4         FileType fileType = getFileType(filePath);
5         System.out.println(fileType.getKey());
6     }
```

上面的输出为 **PDF**，说明我们的实验是成功的，所以我们可以根据魔数在校验文件的类型了。

在经过一顿操作猛如虎之后，看着屏幕的代码，你感到无比的欣慰，原来一个如此简单的需求也是要考虑很多细节的，不能太随意。

提交了代码，关上了电脑，想着自己加班这么晚，真是累，但是看着窗外灯火通明，路上车水马龙，知道还有很多人跟你一样在奋斗，瞬间又有了动力~

## 03、总结

今天通过一个真实的业务场景给大家介绍了编程中魔数的相关应用，希望能帮助到大家。

另外从另一个方面也告诉了我们在遇到一个需求的时候，需要的考虑一点，有时候并不是自己想的那样，需要从一开始就考虑的多一点，这样后面就会省心好多。

最后欢迎大家到我们《Java 极客技术》知识星球中一起谈论技术人生，互相成长，互相进步。

精彩回顾：

重温HTTP，你到底做了什么？

羞，Java 字符串拼接竟然有这么多姿势

带你走入 Flink 的世界

< END >

我们共同组建了一个全网质量最高，但价格最低的知识星球，只需 50 元就可以包养我们一整年，还等什么，快来扫码加入我们！



[阅读原文](#)