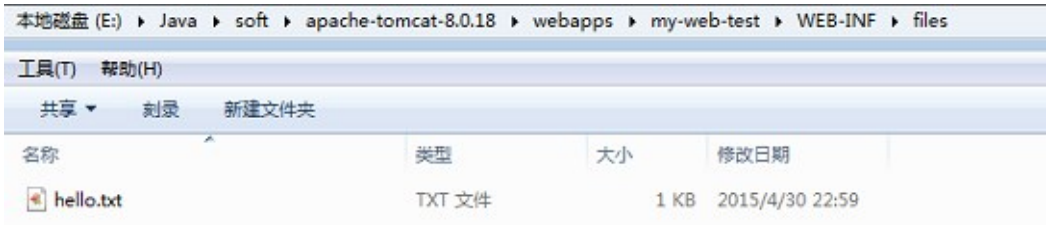
嗯,看到标题,你应该知道本文的主题是啥了.ヽ(ﾟ▽ﾟ)ﾉ, Jetty 的文件上传存在的坑及其多个版本中问题.作为比较,会先以 Tomcat 做测试.这里我们使用 servlet 规范中的 `MultipartConfig` 、 `Part` 等 API.测试程序如下(此例子稍微改了下 《servlet和jsp学习指南》中的示例):

```
@WebServlet(urlPatterns = {"/singleUpload"})
@MultipartConfig
public class SingleUploadServlet extends HttpServlet {
    private String getFilename(Part part) {
        String contentDispositionHeader = part.getHeader("content-disposition");
        String[] elements = contentDispositionHeader.split(";");
        for (String element : elements) {
            if (element.trim().startsWith("filename")) {
                return element.substring(element.indexOf('=') + 1).trim().replace("\"", "")
;
            }
        }

        return null;
    }

    @Override
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws Ser
vletException, IOException {
        Part part = request.getPart("filename");
        String fileName = getFilename(part);
        if (fileName != null && !fileName.isEmpty()) {
            // 保存到 webapp 的 /WEB-INF/files/ 目录中
            String file = getServletContext().getRealPath("/WEB-INF") + File.separator + "f
iles" + File.separator + fileName;
            part.write(file);
        }

        response.setContentType("text/html");
        PrintWriter writer = response.getWriter();
        writer.print("<br/>Uploaded file name: " + fileName);
        writer.print("<br/>Size: " + part.getSize());

        String author = request.getParameter("author");
        writer.print("<br/>Author: " + author);
    }
}
```

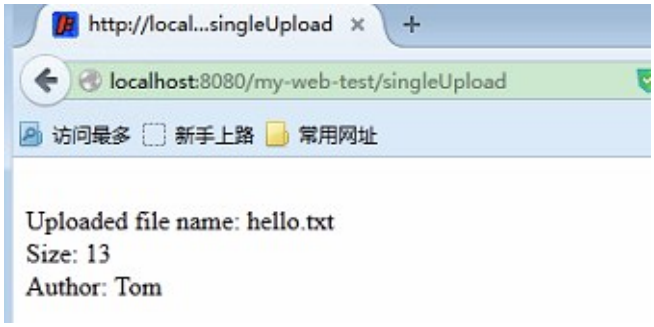
首先以 Tomcat 测试,可以看到保存到了指定的目录:



接下来测试 Jetty, 分别测试三个版本, `jetty-distribution-9.2.3.v20140905` 、 `jetty-distribution-9.2.7.v20150116` 、 `jetty-distribution-9.2.10.v20150310` .

jetty-distribution-9.2.3.v20140905

上传成功,显示了如下页面.



但是在 webapp 的 `WEB-INF/files` 中并没有找到相应的文件.文件没有上传成功? 不是, 文件确实上传成功了, 但是 Jetty 是将它临时保存到 `tempdir` 目录中.这里的 `MultiPart3279539292208376648` 也就是我们的 `hello.txt` 文件.

具体是啥原因导致没有保存到我们期望的目录呢?, 看下 `part.write()` 的代码:

```
1.   public void write(String fileName) throws IOException {
2.       if (_file == null) {
3.           ...
4.       } else {
5.           //the part data is already written to a temporary file, just rename it
6.           _temporary = false;
7.
8.           File f = new File(_tmpDir, fileName);
9.           if (_file.renameTo(f))
10.              _file = f;
11.      }
12.  }
```

看到了吧, 这里使用了 `File.renameTo()` 方法,但是这个方法太不靠谱了.连 API 中也说明了.修正这个问题也比较简单, 不要使用不靠谱的 `File.renameTo()` 方法,使用 `commons-io` 包或 JDK 中的 `Files.move()` 方法将这个临时文件拷贝到期望的目录.但是这里仍然存在一个问题,如果文件较大,从临时目录拷贝到另一个目录也耗时.

好了,到这里貌似已经看到了 Jetty 文件上传的坑了,应该可以结束了,不过一开始其实并没有看具体的源码.而是先测试了两个高版本.然后发现了一些悲催的事情.接着往下看.

jetty-distribution-9.2.7.v20150116

这个版本没有跟踪代码,ヽ(ﾟ▽ﾟ)ﾉ.

jetty-distribution-9.2.10.v20150310

```
2015-04-30 23:22:43.094:WARN:oejs.ServletHandler:qtp1567581361-15: /my-web-test/singleUpload
java.io.FileNotFoundException: E:\Java\soft\jetty-distribution-9.2.10.v20150310\my-demo-base\work\jetty-0.0.0.0-8080-my-web-test-_my-web-test-any-\E:\Java\soft\jetty-distribution-9.2.10.v20150310\my-demo-base\webapps\my-web-test\WEB-INF\files\hello.txt (文件名、目录名或卷标语法不正确。)
        at java.io.FileOutputStream.open0(Native Method)
        at java.io.FileOutputStream.open(FileOutputStream.java:270)
        at java.io.FileOutputStream.<init>(FileOutputStream.java:213)
        at java.io.FileOutputStream.<init>(FileOutputStream.java:162)
        at org.eclipse.jetty.util.MultiPartInputStreamParser$MultiPart.write(MultiPartInputStreamParser.java:260)
        at file.SingleUploadServlet.doPost(SingleUploadServlet.java:64)
```

哎, 看到这儿是不是觉得 Jetty 这部分挺悲催的.追了下这部分的代码,发现是因为 `MultiPartInputStreamParser.MultiPart#open()` 方法移除了以下的代码所导致的:

```
1.    if (_filename != null && _filename.trim().length() > 0) {
2.        createFile();
3.    }
```

9.2.3

```
1.    protected void open() throws IOException {
2.        if (_filename != null && _filename.trim().length() > 0) {
3.            createFile();
4.        } else {
5.            //Write to a buffer in memory until we discover we've exceed the
6.            //MultipartConfig fileSizeThreshold
7.            _out = _bout = new ByteArrayOutputStream2();
8.        }
9.    }
```

9.2.10

```
1.    protected void open() throws IOException {
2.        //Write to a buffer in memory until we discover we've exceed the
3.        //MultipartConfig fileSizeThreshold
4.        _out = _bout= new ByteArrayOutputStream2();
5.    }
```

Jetty 自身的文件上传还是存在不少问题的, 且用且珍惜.