POST一个多部分编码(Multipart-Encoded)的文件¶

Requests 使得上传多部分编码文件变得很简单：

```
>>> url = 'http://httpbin.org/post'
>>> files = {'file': open('report.xls', 'rb')}

>>> r = requests.post(url, files=files)
>>> r.text
{
  ...
  "files": {
    "file": "<censored...binary...data>"
  },
  ...
}
```

你可以显式地设置文件名，文件类型和请求头：

```
>>> url = 'http://httpbin.org/post'
>>> files = {'file': ('report.xls', open('report.xls', 'rb'), 'application/vnd.ms-excel', {'Expires': '0'})}

>>> r = requests.post(url, files=files)
>>> r.text
{
  ...
  "files": {
    "file": "<censored...binary...data>"
  },
  ...
}
```

如果你想，你也可以发送作为文件来接收的字符串：

```
>>> url = 'http://httpbin.org/post'
>>> files = {'file': ('report.csv', 'some,data,to,send\nanother,row,to,send\n')}

>>> r = requests.post(url, files=files)
>>> r.text
{
  ...
  "files": {
    "file": "some,data,to,send\\nanother,row,to,send\\n"
  },
  ...
}
```

如果你发送一个非常大的文件作为 `multipart/form-data` 请求，你可能希望将请求做成数据流。默认下 `requests` 不支持，但有个第三方包 `requests-toolbelt` 是支持的。你可以阅读 toolbelt 文档来了解使用方法。

在一个请求中发送多文件参考 高级用法 一节。

警告

我们强烈建议你用二进制模式(binary mode)打开文件。这是因为 Requests 可能会试图为你提供 `Content-Length` header，在它这样做的时候，这个值会被设为文件的字节数（*bytes*）。如果用文本模式(text mode)打开文件，就可能会发生错误。

响应状态码

我们可以检测响应状态码：

```
>>> r = requests.get('http://httpbin.org/get')
>>> r.status_code
200
```

为方便引用，Requests还附带了一个内置的状态码查询对象：

```
>>> r.status_code == requests.codes.ok
True
```

如果发送了一个错误请求(一个 4XX 客户端错误，或者 5XX 服务器错误响应)，我们可以通过**`Response.raise_for_status()`** 来抛出异常：

```
>>> bad_r = requests.get('http://httpbin.org/status/404')
>>> bad_r.status_code
404

>>> bad_r.raise_for_status()
Traceback (most recent call last):
  File "requests/models.py", line 832, in raise_for_status
    raise http_error
requests.exceptions.HTTPError: 404 Client Error
```

但是，由于我们的例子中 `r` 的 `status_code` 是 `200` ，当我们调用 `raise_for_status()` 时，得到的是：

```
>>> r.raise_for_status()
None
```

一切都挺和谐哈。