1.3响应头和响应码

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。

响应头

我们可以查看以一个 Python 字典形式展示的服务器响应头：

```
>>> r.headers
{
    'content-encoding': 'gzip',
    'transfer-encoding': 'chunked',
    'connection': 'close',
    'server': 'nginx/1.0.4',
    'x-runtime': '148ms',
    'etag': '"e1ca502697e5c9317743dc078f67693f"',
    'content-type': 'application/json'
}
```

但是这个字典比较特殊：它是仅为 HTTP 头部而生的。根据 RFC 2616， HTTP 头部是大小写不敏感的。

因此，我们可以使用任意大写形式来访问这些响应头字段：

```
>>> r.headers['Content-Type']
'application/json'

>>> r.headers.get('content-type')
'application/json'
```

它还有一个特殊点，那就是服务器可以多次接受同一 header，每次都使用不同的值。但 Requests 会将它们合并，这样它们就可以用一个映射来表示出来，参见 RFC 7230:

> A recipient MAY combine multiple header fields with the same field name into one "field-name: field-value" pair, without changing the semantics of the message, by appending each subsequent field value to the combined field value in order, separated by a comma.
>
> 接收者可以合并多个相同名称的 header 栏位，把它们合为一个 "field-name: field-value" 配对，将每个后续

的栏位值依次追加到合并的栏位值中，用逗号隔开即可，这样做不会改变信息的语义。