

HenCoder Plus 第 1 课 讲义

HTTP 的概念、原理、工作机制、数据格式

HTTP 的定义

一种网络传输协议，位于 TCP / IP 协议族的最顶层——应用层（这句话后面的课会详细解释）。

HTTP

Hypertext Transfer Protocol，超文本传输协议，和 HTML (Hypertext Markup Language 超文本标记语言) 一起诞生，用于在网络上请求和传输 HTML 内容。

超文本，即「扩展型文本」，指的是 HTML 中可以有链向别的文本的链接（hyperlink）。

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

This is a Heading

This is a paragraph.

HTTP 的工作方式

浏览器：

用户输入地址后回车或点击链接 -> 浏览器拼装 HTTP 报文并发送请求给服务器 -> 服务器处理请求后发送响应报文给浏览器 -> 浏览器解析响应报文并使用渲染引擎显示到界面

手机 App：

用户点击或界面自动触发联网需求 -> Android 代码调用拼装 HTTP 报文并发送请求到服务器 -> 服务器处理请求后发送响应报文给手机 -> Android 代码处理响应报文并作出相应处理（如储存数据、加工数据、显示数据到界面）

URL 和 HTTP 报文

URL 格式

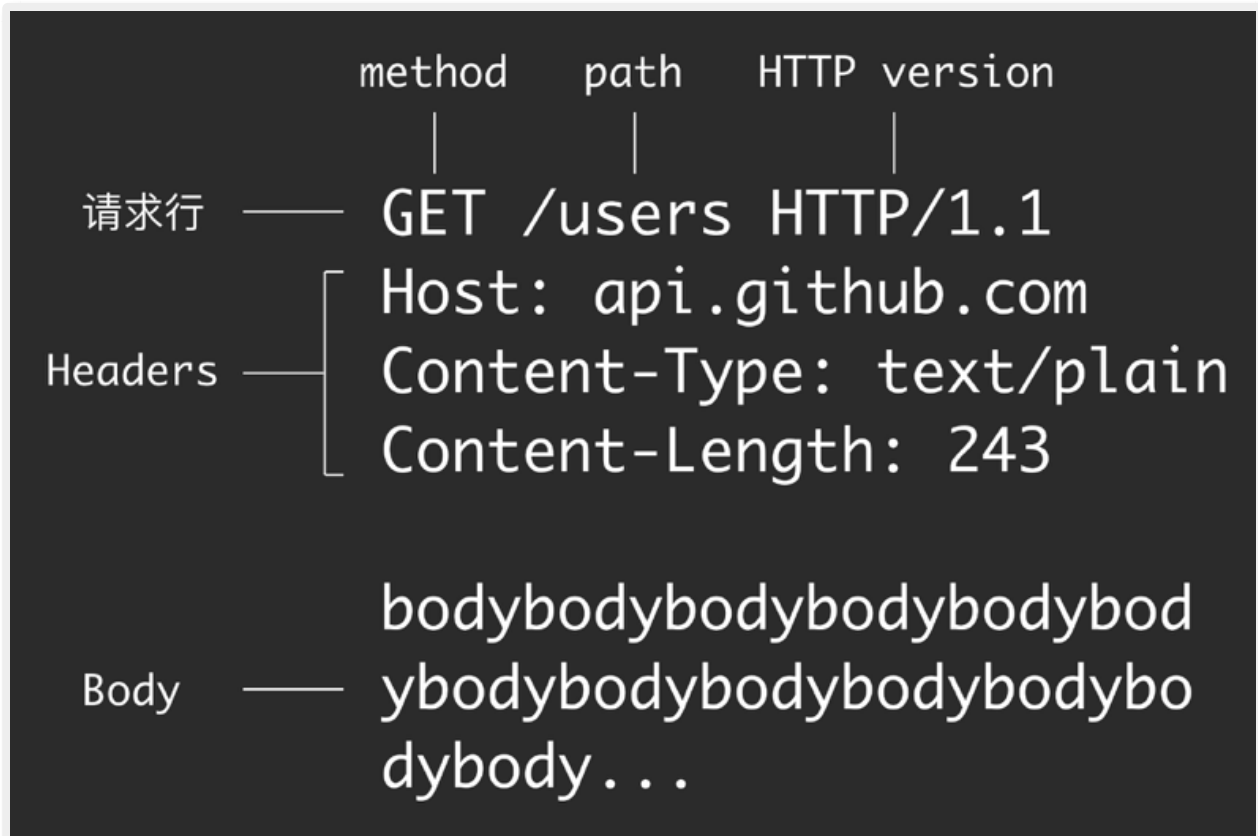
三部分：协议类型、服务器地址(和端口号)、路径(Path)

协议类型://服务器地址[:端口号]路径

<http://hencoder.com/users?gender=male>

报文格式

请求报文



响应报文



Request Method 请求方法

GET

- 用于获取资源
- 对服务器数据不进行修改
- 不发送 Body

```
GET /users/1 HTTP/1.1
Host: api.github.com
```

对应 Retrofit 的代码：

```
@GET("/users/{id}")
Call<User> getUser(@Path("id") String id, @Query("gender") String gender);
```

POST

- 用于增加或修改资源
- 发送给服务器的内容写在 Body 里面

```
POST /users HTTP/1.1
Host: api.github.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

name=rengwuxian&gender=male
```

对应 Retrofit 的代码：

```
@FormUrlEncoded
@POST("/users")
Call<User> addUser(@Field("name") String name, @Field("gender") String gender);
```

PUT

- 用于修改资源
- 发送给服务器的内容写在 Body 里面

```
PUT /users/1 HTTP/1.1
Host: api.github.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

gender=female
```

对应 Retrofit 的代码：

```
@FormUrlEncoded
@PUT("/users/{id}")
Call<User> updateGender(@Path("id") String id, @Field("gender") String
gender);
```

DELETE

- 用于删除资源
- 不发送 Body

```
DELETE /users/1 HTTP/1.1
Host: api.github.com
```

```
@DELETE("/users/{id}")
Call<User> getUser(@Path("id") String id, @Query("gender") String gender);
```

HEAD

- 和 GET 使用方法完全相同
- 和 GET 唯一区别在于，返回的响应中没有 Body

Status Code 状态码

三位数字，用于对响应结果做出类型化描述（如「获取成功」「内容未找到」）。

- 1xx：临时性消息。如：100（继续发送）、101（正在切换协议）
- 2xx：成功。最典型的是 200（OK）、201（创建成功）。
- 3xx：重定向。如 301（永久移动）、302（暂时移动）、304（内容未改变）。
- 4xx：客户端错误。如 400（客户端请求错误）、401（认证失败）、403（被禁止）、404（找不到内容）。
- 5xx：服务器错误。如 500（服务器内部错误）。

Header 首部

作用：HTTP 消息的 metadata。

Host

目标主机。注意：不是在网络上用于寻址的，而是在目标服务器上用于定位子服务器的。

Content-Type

指定 Body 的类型。主要有四类：

1. text/html

请求 Web 页面是返回响应的类型，Body 中返回 html 文本。格式如下：

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 853

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  .....
```

2. x-www-form-urlencoded

Web 页面纯文本表单的提交方式。

Name

昵称

```
<form method="POST" enctype="application/x-www-form-urlencoded">
  <div class="form-group">
    <label>Name</label>
    <input type="text" name="name" class="form-control" placeholder="姓名">
  </div>
  <div class="form-group">
    <label>昵称</label>
    <input type="text" name="nickName" class="form-control" placeholder="昵称">
  </div>
  <p>
    <button type="submit" class="btn btn-primary btn-lg">提交</button>
  </p>
</form>
```

格式如下：

```
POST /users HTTP/1.1
Host: api.github.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 27

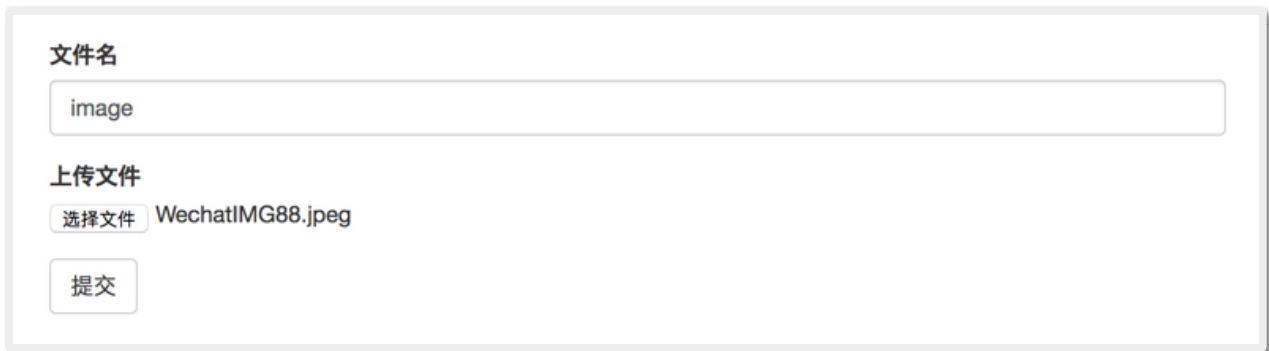
name=rengwuxian&gender=male
```

对应 Retrofit 的代码：

```
@FormUrlEncoded
@POST("/users")
Call<User> addUser(@Field("name") String name, @Field("gender") String
gender);
```

3. multipart/form-data

Web 页面含有二进制文件时的提交方式。



```
<form method="POST" enctype="multipart/form-data">
  <div class="form-group">
    <label>文件名</label>
    <input type="text" class="form-control" name="name" id="name" placeholder="文件名">
  </div>
  <div class="form-group">
    <label for="exampleInputFile">上传文件</label>
    <input type="file" name="upload" id="exampleInputFile">
  </div>
  <button type="submit" class="btn btn-default">提交</button>
</form>
```

格式如下：

```
POST /users HTTP/1.1
Host: hencoder.com
Content-Type: multipart/form-data; boundary=----
WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Length: 2382
```

```
-----WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="name"

rengwuxian
-----WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="avatar"; filename="avatar.jpg"
Content-Type: image/jpeg

JFIFHHv0wX9jximQrWa.....
-----WebKitFormBoundary7MA4YWxkTrZu0gW--
```

对应 Retrofit 的代码:

```
@Multipart
@POST("/users")
Call<User> addUser(@Part("name") RequestBody name, @Part("avatar")
RequestBody avatar);

...

RequestBody namePart = RequestBody.create(MediaType.parse("text/plain"),
nameStr);
RequestBody avatarPart = RequestBody.create(MediaType.parse("image/jpeg"),
avatarFile);
api.addUser(namePart, avatarPart);
```

4. application/json , image/jpeg , application/zip ...

单项内容（文本或非文本都可以），用于 Web Api 的响应或者 POST / PUT 的请求

请求中提交 JSON

```
POST /users HTTP/1.1
Host: hencoder.com
Content-Type: application/json; charset=utf-8
Content-Length: 38

{"name":"rengwuxian","gender":"male"}
```

对应 Retrofit 的代码:

```
@POST("/users")
Call<User> addUser(@Body("user") User user);

...

// 需要使用 JSON 相关的 Converter
api.addUser(user);
```

响应中返回 JSON

```
HTTP/1.1 200 OK
content-type: application/json; charset=utf-8
content-length: 234

[{"login":"mojombo","id":1,"node_id":"MDQ6VXNl
cjE=","avatar_url":"https://avatars0.githubuse
rcontent.com/u/1?v=4","gravatar.....
```

请求中提交二进制内容

```
POST /user/1/avatar HTTP/1.1
Host: hencoder.com
Content-Type: image/jpeg
Content-Length: 1575

JFIFHH9.....
```

对应 Retrofit 的代码：

```
@POST("users/{id}/avatar")
Call<User> updateAvatar(@Path("id") String id, @Body RequestBody avatar);

...

RequestBody avatarBody = RequestBody.create(MediaType.parse("image/jpeg"),
avatarFile);
api.updateAvatar(id, avatarBody)
```

相应中返回二进制内容

```
HTTP/1.1 200 OK
content-type: image/jpeg
content-length: 1575

JFIFHH9.....
```

Content-Length

指定 Body 的长度（字节）。

Transfer: chunked (分块传输编码 Chunked Transfer Encoding)

用于当响应发起时，内容长度还没能确定的情况下。和 Content-Length 不同时使用。用途是尽早给出响应，减少用户等待。

格式：

```
HTTP/1.1 200 OK
Content-Type: text/html
Transfer-Encoding: chunked

4
Chun
9
ked Trans
12
fer Encoding
0
```

Location

指定重定向的目标 URL

User-Agent

用户代理，即是谁实际发送请求、接受响应的，例如手机浏览器、某款手机 App。

Range / Accept-Range

按范围取数据

`Accept-Range: bytes` 响应报文中出现，表示服务器支持按字节来取范围数据

`Range: bytes=<start>-<end>` 请求报文中出现，表示要取哪段数据

`Content-Range:<start>-<end>/total` 响应报文中出现，表示发送的是哪段数据

作用：断点续传、多线程下载。

其他 Headers

- Accept: 客户端能接受的数据类型。如 text/html
- Accept-Charset: 客户端接受的字符集。如 utf-8
- Accept-Encoding: 客户端接受的压缩编码类型。如 gzip
- Content-Encoding: 压缩类型。如 gzip

Cache

作用：在客户端或中间网络节点缓存数据，降低从服务器取数据的频率，以提高网络性能。

REST

REST 的定义众说纷纭，没有统一答案。

扔物线的观点：REST HTTP 即正确使用 HTTP。包括：

- 使用资源的格式来定义 URL
- 规范地使用 method 来定义网络请求操作
- 规范地使用 status code 来表示响应状态
- 其他符合 HTTP 规范的设计准则