网络原理之Http

> - 掌握`http`原理，重点掌握`http Request & Response`格式
> - 掌握`http`中相关重点知识，如请求方法，属性，状态码等
> - 使用`java socket`编写简易版本`http server`, 深刻理解原理
> - 掌握session和cookie
> - 编写简单的`html`，感受前后端语言差别

1. `Http`原理

1.1 理解为何要有应用层?

我们已经学过`TCP/IP`, 已经知道目前数据能从客户端进程经过路径选择跨网络传送到服务器端进程[`IP+Port`],可是，仅仅把数据从A点传送到B点就完了吗？这就好比，在淘宝上买了一部手机，卖家[客户端]把手机通过顺丰[传送+路径选择] 送到买家 [服务器] 手里就完了吗？当然不是，买家还要使用这款产品，还要在使用之后，给卖家打分评论。所以，我们把数据从A端传送到B端，`TCP/IP`解决的是顺丰的功能，而两端还要对数据进行加工处理或者使用，所以我们还需要一层协议，不关心通信细节，关心应用细节!

这层协议叫做应用层协议。而应用是有不同的场景的，所以应用层协议是有不同种类的，其中经典协议之一的HTTP就是其中的佼佼者。那么，`Http`是解决什么应用场景呢?

早期用户，上网使用浏览器来进行上网，而用浏览器上网阅读信息，最常见的是查看各种网页【其实也是文件数据，不过是一系列的`html`文档，当然还有其他资源如图片，`css`，`js`等】,而要把网页文件信息通过网络传送到客户端，或者把用户数据上传到服务器，就需要`Http`协议【当然，http作用不限于此】

1.2 再谈 "协议"

那如何理解应用层协议呢？再回到我们刚刚说的买手机的例子，顺丰相当于`TCP/IP`的功能，那么买回来的手机都附带了说明书【产品介绍，使用介绍，注意事项等】，而该说明书指导用户该如何使用手机【虽然我们都不看，但是父母辈有部分是有看说明书的习惯的：）】，此时的说明书可以理解为用户层协议

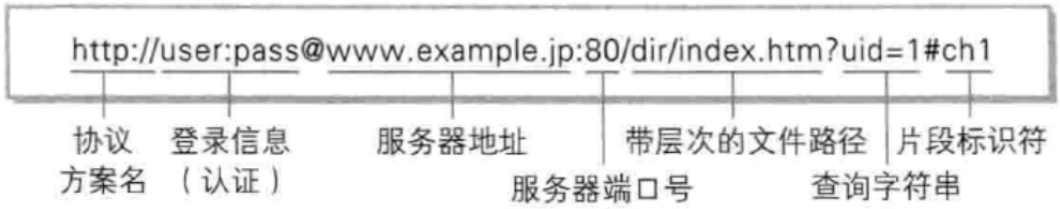
1.3 HTTP协议

虽然我们说, 应用层协议是我们程序猿自己定的.

但实际上, 已经有大佬们定义了一些现成的, 又非常好用的应用层协议, 供我们直接参考使用. HTTP(超文本传输协议)就是其中之一.

1.4 认识URL

平时我们俗称的 "网址" 其实就是说的 URL



1.5 urlencode和urldecode

像 / ? : 等这样的字符, 已经被url当做特殊意义理解了. 因此这些字符不能随意出现.

比如, 某个参数中需要带有这些特殊字符, 就必须先对特殊字符进行转义.

转义的规则如下:

将需要转码的字符转为16进制，然后从右到左，取4位(不足4位直接处理)，每2位做一位，前面加上%，编码成%XY格式

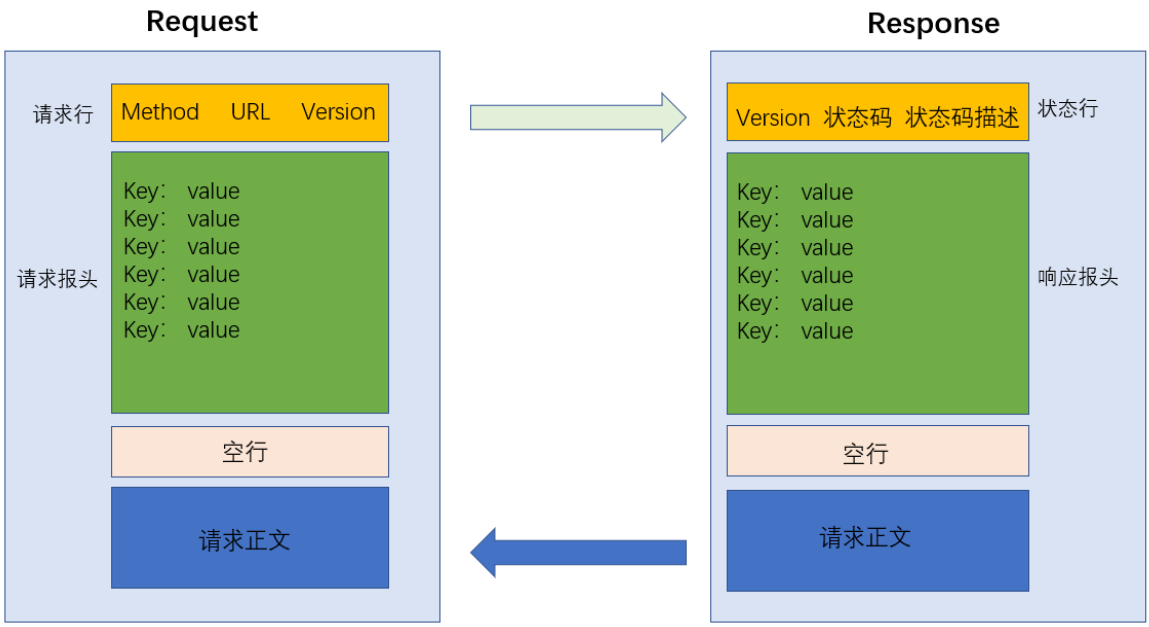
例如:

"+" 被转义成了 "%2B"

urldecode就是urlencode的逆过程;

urlencode工具

1.6 HTTP协议格式



HTTP请求

```
POST http://job.xjtu.edu.cn/companyLogin.do HTTP/1.1
Host: job.xjtu.edu.cn
Connection: keep-alive
Content-Length: 36
Cache-Control: max-age=0
Origin: http://job.xjtu.edu.cn
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/61.0.3163.100 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/
apng,*/*;q=0.8
Referer: http://job.xjtu.edu.cn/companyLogin.do
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
Cookie: JSESSIONID=D628A75845A74D29D991DB47A461E4FC;
Hm_lvt_783e83ce0ee350e23a9d389df580f658=1504963710,1506661798;
Hm_lpvt_783e83ce0ee350e23a9d389df580f658=1506661802

username=hgtz2222&password=222222222
```

- 首行: [方法] + [url] + [版本]
- Header: 请求的属性, 冒号分割的键值对;每组属性之间使用\n分隔;遇到空行表示Header部分结束
- Body: 空行后面的内容都是Body. Body允许为空字符串. 如果Body存在, 则在Header中会有一个Content-Length属性来标识Body的长度;

HTTP响应

```
HTTP/1.1 200 OK
Server: YxlinkWAF
Content-Type: text/html;charset=UTF-8
Content-Language: zh-CN
Transfer-Encoding: chunked
Date: Fri, 29 Sep 2017 05:10:13 GMT

<!DOCTYPE html>
<html>
<head>
<title>西安交通大学就业网</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="IE=Edge">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="shortcut icon" href="/renovation/images/icon.ico">
<link href="/renovation/css/main.css" rel="stylesheet" media="screen" />
<link href="/renovation/css/art_default.css" rel="stylesheet" media="screen" />
<link href="/renovation/css/font-awesome.css" rel="stylesheet" media="screen" />
<script type="text/javascript" src="/renovation/js/jquery1.7.1.min.js"></script>
<script type="text/javascript" src="/renovation/js/main.js"></script><!--main-->
<link href="/style/warmTipsstyle.css" rel="stylesheet" type="text/css">
</head>
```

- 首行: [版本号] + [状态码] + [状态码解释]
- Header: 请求的属性, 冒号分割的键值对;每组属性之间使用\n分隔;遇到空行表示Header部分结束
- Body: 空行后面的内容都是Body. Body允许为空字符串. 如果Body存在, 则在Header中会有一个Content-Length属性来标识Body的长度; 如果服务器返回了一个html页面, 那么html页面内容就是在body中.

1.7 HTTP的方法

方法	说明	支持的HTTP协议版本
GET	获取资源	1.0、1.1
POST	传输实体主体	1.0、1.1
PUT	传输文件	1.0、1.1
HEAD	获得报文首部	1.0、1.1
DELETE	删除文件	1.0、1.1
OPTIONS	询问支持的方法	1.1
TRACE	追踪路径	1.1
CONNECT	要求用隧道协议连接代理	1.1
LINK	建立和资源之间的联系	1.0
UNLINE	断开连接关系	1.0

其中最常用的就是GET方法和POST方法.

1.8 HTTP的状态码

	类别	原因短语
1XX	Informational（信息性状态码）	接收的请求正在处理
2XX	Success（成功状态码）	请求正常处理完毕
3XX	Redirection（重定向状态码）	需要进行附加操作以完成请求
4XX	Client Error（客户端错误状态码）	服务器无法处理请求
5XX	Server Error（服务器错误状态码）	服务器处理请求出错

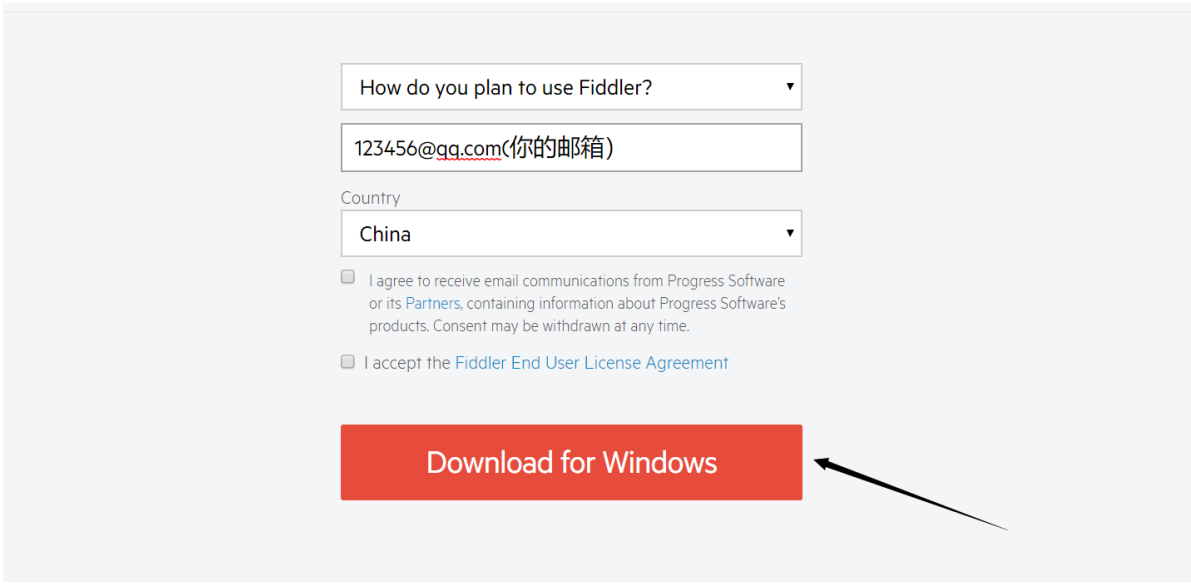
最常见的状态码, 比如 200(OK), 404(Not Found), 403(Forbidden), 302(Redirect, 重定向), 504(Bad Gateway)

1.9 HTTP常见Header

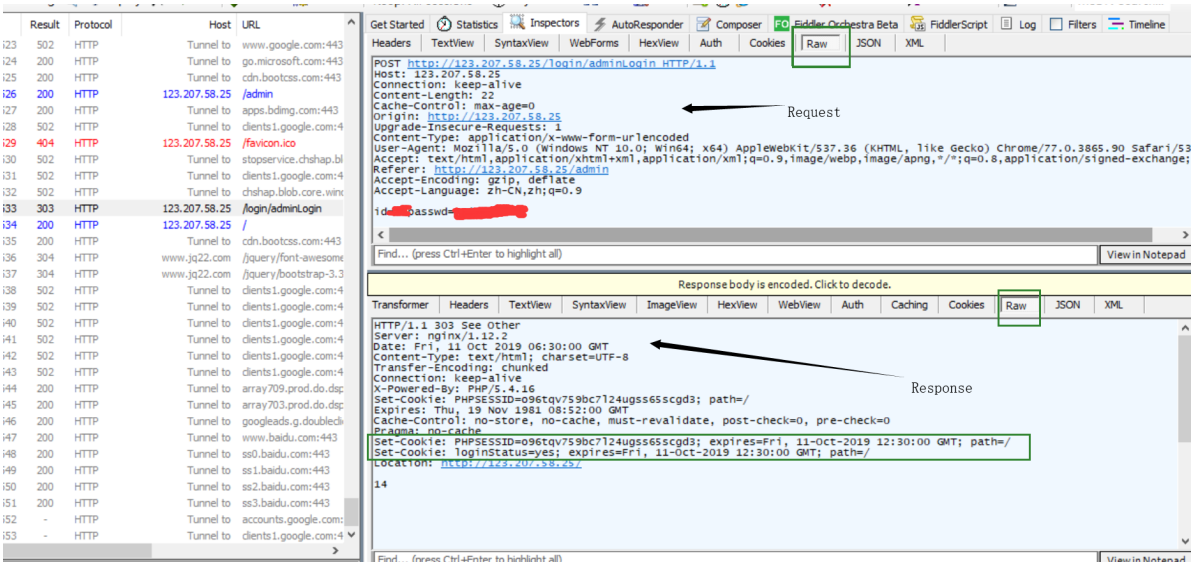
> - Content-Type: 数据类型(text/html等)
> - Content-Length: Body的长度
> - Host: 客户端告知服务器, 所请求的资源是在哪个主机的哪个端口上;
> - User-Agent: 声明用户的操作系统和浏览器版本信息;
> - referer: 当前页面是从哪个页面跳转过来的;
> - location: 搭配3xx状态码使用, 告诉客户端接下来要去哪里访问;
> - Cookie: 用于在客户端存储少量信息. 通常用于实现会话(session)的功能;

1.10. 抓包工具 fiddler

为了研究http协议相关协议细节【主要是request和response，状态码，以及后续的cookie和session】，我们引进一款抓包工具fiddler，同学们可以去官网下载安装：https://www.telerik.com/fiddler/



下载好之后，打开之后，就可以直接进行使用了



同学们可以根据实际情况，演示查看其他功能，比如重定向，404，上传文件等

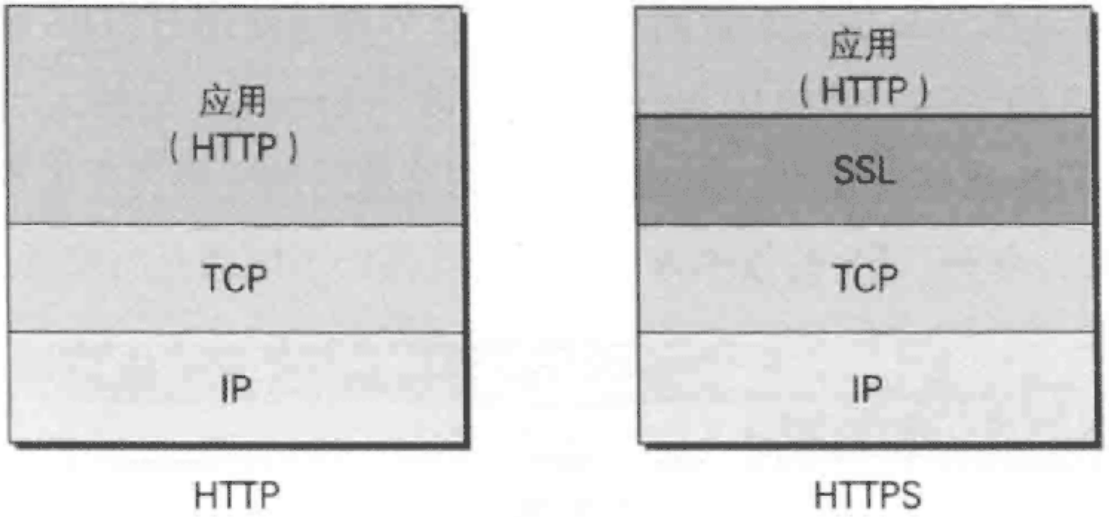
1.11. 其他

User-Agent里的历史故事

2. 补充主题

2.1. `Http` VS `Https`

简单理解,http传送数据（包括账号和密码），都是明文传送，很容易被窃取或者侦听，在现有的互联网应用中，很明显有不安全因素，所以有了https，可以简单理解成https多了一层加密解密层，在发送前加密，在收到后解密，在网络里传输的都是经过加密的数据



详细介绍：https://blog.csdn.net/xiaoming100001/article/details/81109617

https://www.cnblogs.com/Rawls/p/10725703.html

2.2. `Http` 新特性

- http2.0
- http3.0

2.3. 市场上常见的`Http`服务器

- Tomcat(web 应用服务器)
- Nginx
- Apache
- Lighttpd