web 页面解析的流程学习

笔记本:	信安之路学习历程		
创建时间:	2019/10/14 22:39	**更新时间:**	2019/10/18 0:33
作者:	sxwnysh@163.com		
URL:	https://dailc.github.io/2018/03/12/whenyouenteraurl.html		

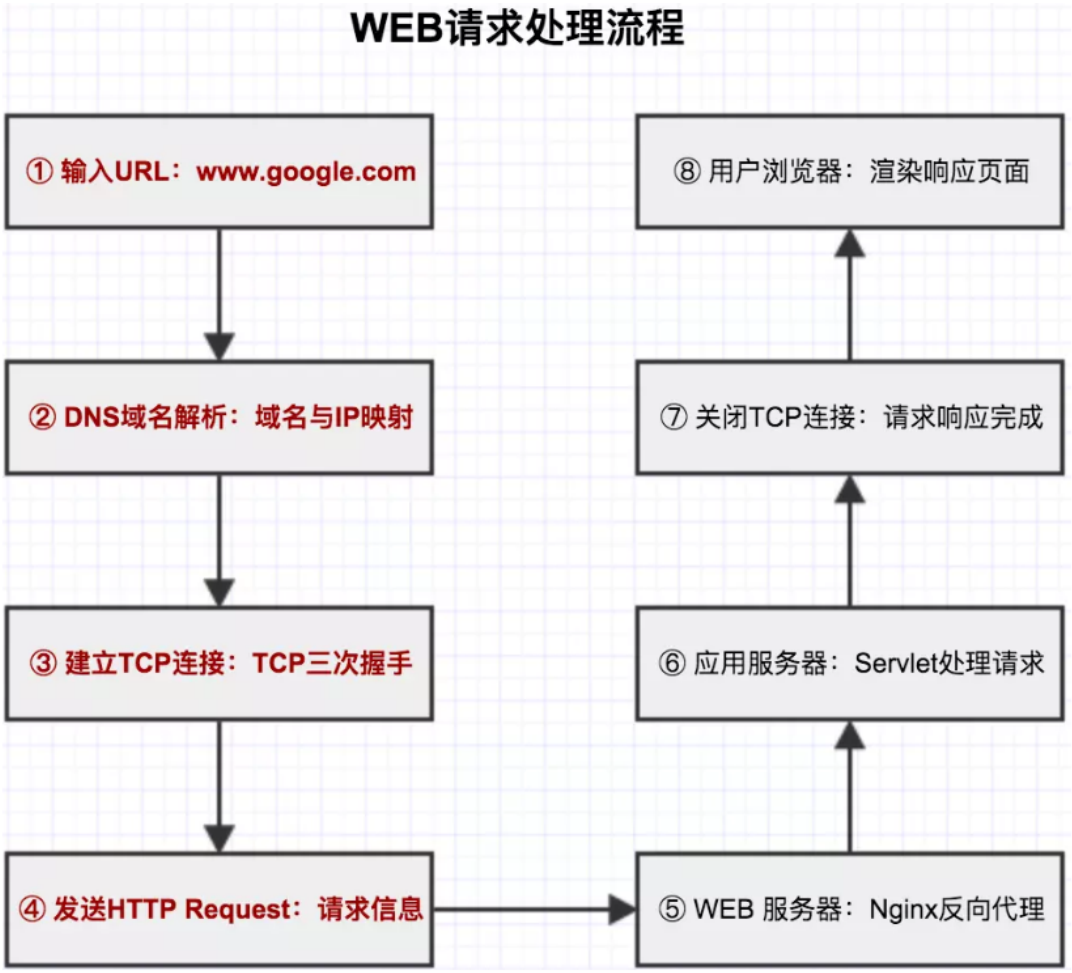
web 页面解析的流程学习

从输入URL到页面加载发生了什么

总体来说分为以下几个过程:

> **1. DNS解析**
>
> **2. TCP连接**
>
> **3. 发送HTTP请求**
>
> **4. 服务器处理请求并返回HTTP报文**
>
> **5. 浏览器解析渲染页面**
>
> **6. 连接结束**

为了直观明了，先上一张图



一、DNS解析流程

- **DNS解析步骤**

> **1.** 在浏览器中输入www.qq.com域名，操作系统会先检查自己本地的hosts文件是否有这个网址映射关系，如果有，就先调用这个IP地址映射，完成域名解析。
>
> **2.** 如果hosts里没有这个域名的映射，则查找本地DNS解析器缓存，是否有这个网址映射关系，如果有，直接返回，完成域名解析。
>
> **3.** 如果hosts与本地DNS解析器缓存都没有相应的网址映射关系，首先会找TCP/ip参数中设置的首选DNS服务器，在此我们叫它本地DNS服务器，此服务器收到查询时，如果要查询的域名，包含在本地配置区域资源中，则返回解析结果给客户机，完成域名解析，此解析具有权威性

> **4.** 如果要查询的域名，不由本地DNS服务器区域解析，但该服务器已缓存了此网址映射关系，则调用这个IP地址映射，完成域名解析，此解析不具有权威性。
>
> **5.** 如果本地DNS服务器本地区域文件与缓存解析都失效，则根据本地DNS服务器的设置（是否设置转发器）进行查询，如果未用转发模式，本地DNS就把请求发至13台根DNS，根DNS服务器收到请求后会判断这个域名(.com)是谁来授权管理，并会返回一个负责该顶级域名服务器的一个IP。本地DNS服务器收到IP信息后，将会联系负责.com域的这台服务器。这台负责.com域的服务器收到请求后，如果自己无法解析，它就会找一个管理.com域的下一级DNS服务器地址(qq.com)给本地DNS服务器。当本地DNS服务器收到这个地址后，就会找qq.com域服务器，重复上面的动作，进行查询，直至找到www.qq.com主机。
>
> **6.** 如果用的是转发模式，此DNS服务器就会把请求转发至上一级DNS服务器，由上一级服务器进行解析，上一级服务器如果不能解析，或找根DNS或把转请求转至上上级，以此循环。不管是本地DNS服务器用是是转发，还是根提示，最后都是把结果返回给本地DNS服务器，由此DNS服务器再返回给客户机。

- ## DNS相关安全问题

1. DNS欺骗

DNS欺骗即域名信息欺骗是最常见的DNS安全问题。当一个DNS服务器掉入陷阱，使用了来自一个恶意DNS服务器的错误信息，那么该DNS服务器就被欺骗了。DNS欺骗会使那些易受攻击的DNS服务器产生许多安全问题，例如：将用户引导到错误的互联网站点，或者发送一个电子邮件到一个未经授权的邮件服务器。网络攻击者通常通过两种方法进行DNS欺骗。

> **缓存感染：黑客会熟练的使用DNS请求，将数据放入一个没有设防的DNS服务器的缓存当中。这些缓存信息会在客户进行DNS访问时返回给客户，从而将客户引导到入侵者所设置的运行木马的Web服务器或邮件服务器上，然后黑客从这些服务器上获取用户信息。**
>
> **DNS信息劫持：入侵者通过监听客户端和DNS服务器的对话，通过猜测服务器响应给客户端的DNS查询ID。每个DNS报文包括一个相关联的16位ID号，DNS服务器根据这个ID号获取请求源位置。黑客在DNS服务器之前将虚假的响应交给用户，从而欺骗客户端去访问恶意的网站。DNS重定向：攻击者能够将DNS名称查询重定向到恶意DNS服务器。这样攻击者可以获得DNS服务器的写权限**

2. 拒绝服务攻击

黑客主要利用一些DNS软件的漏洞，如在BIND 9版本（版本9.2.0以前的 9系列）如果有人向运行BIND的设备发送特定的DNS数据包请求，BIND就会自动关闭。攻击者只能使BIND关闭，而无法在服务器上执行任意命令。如果得不到DNS服务，那么就会产生一场灾难：由于网址不能解析为IP地址，用户将无方访问互联网。这样，DNS产生的问题就好像是互联网本身所产生的问题，这将导致大量的混乱。

3. 分布式拒绝服务攻击

DDOS 攻击通过使用攻击者控制的几十台或几百台计算机攻击一台主机，使得服务拒绝攻击更难以防范，更难以通过阻塞单一攻击源主机的数据流，来防范服务拒绝攻击。

4. 缓冲区漏洞溢出攻击

黑客利用DNS服务器软件存在漏洞，比如对特定的输入没有进行严格检查，那么有可能被攻击者利用，攻击者构造特殊的畸形数据包来对DNS服务器进行缓冲区溢出攻击。如果这一攻击成功，就会造成DNS服务停止，或者攻击者能够在DNS服务器上执行其设定的任意代码。

二、 Tcp/Ip请求

- http的本质就是tcp/ip请求
- 需要了解3次握手规则建立连接以及断开连接时的四次挥手
- tcp将http长报文划分为短报文，通过三次握手与服务端建立连接，进行可靠传输

三次握手的步骤：（抽象派）

> 客户端：hello，你是server么？
> 服务端：hello，我是server，你是client么
> 客户端：yes，我是client

建立连接成功后，接下来就正式传输数据然后，待到断开连接时，需要进行四次挥手（因为是全双工的，所以需要四次挥手）

> 主动方：我已经关闭了向你那边的主动通道了，只能被动接收了
> 被动方：收到通道关闭的信息
> 被动方：那我也告诉你，我这边向你的主动通道也关闭了
> 主动方：最后收到数据，之后双方无法通信

三、五层因特网协议栈

简括就是：从应用层的发送http请求，到传输层通过三次握手建立tcp/ip连接，再到网络层的ip寻址，再到数据链路层的封装成帧，最后到物理层的利用物理介质传输。

当然，服务端的接收就是反过来的步骤

五层因特网协议栈其实就是：

1.应用层(dns,http) DNS解析成IP并发送http请求

2.传输层(tcp,udp) 建立tcp连接（三次握手）

3.网络层(IP,ARP) IP寻址

4.数据链路层(PPP) 封装成帧

5.物理层(利用物理介质传输比特流) 物理传输（然后传输的时候通过双绞线，电磁波等各种介质）

四、从服务器接收到请求到对应后台接收到请求

服务端在接收到请求时，内部会进行很多的处理这里由于不是专业的后端分析，所以只是简单的介绍下，不深入

五、负载均衡

对于大型的项目，由于并发访问量很大，所以往往一台服务器是吃不消的，所以一般会有若干台服务器组成一个集群，然后配合反向代理实现负载均衡当然了，负载均衡不止这一种实现方式，这里不深入...

简单的说：

用户发起的请求都指向调度服务器（反向代理服务器，譬如安装了nginx控制负载均衡），然后调度服务器根据实际的调度算法，分配不同的请求给对应集群中的服务器执行，然后调度器等待实际服务器的HTTP响应，并将它反馈给用户

六、后台处理

一般后台都是部署到容器中的，所以一般为：

> - 先是容器接受到请求（如tomcat容器）
> - 然后对应容器中的后台程序接收到请求（如java程序）
> - 然后就是后台会有自己的统一处理，处理完后响应响应结果

概括下：

> - 一般有的后端是有统一的验证的，如安全拦截，跨域验证
> - 如果这一步不符合规则，就直接返回了相应的http报文（如拒绝请求等）
> - 然后当验证通过后，才会进入实际的后台代码，此时是程序接收到请求，然后执行（譬如查询数据库，大量计算等等）
> - 等程序执行完毕后，就会返回一个http响应包（一般这一步也会经过多层封装）
> - 然后就是将这个包从后端发送到前端，完成交互

七、后台和前台的http交互

http报文结构

报文一般包括了：通用头部，请求/响应头部，请求/响应体

通用头部

> Request Url: 请求的web服务器地址

> Request Method: 请求方式（Get、POST、OPTIONS、PUT、HEAD、DELETE、CONNECT、TRACE）

> Status Code: 请求的返回状态码，如200代表成功

> Remote Address: 请求的远程服务器地址（会转为IP）

Method

> HTTP1.0定义了三种请求方法： GET, POST 和 HEAD方法。以及几种Additional Request Methods：PUT、DELETE、LINK、UNLINK

> HTTP1.1定义了八种请求方法：GET、POST、HEAD、OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法。

状态码(列举几个常见的)

> 200——表明该请求被成功地完成，所请求的资源发送回客户端
> 304——自从上次请求后，请求的网页未修改过，请客户端使用本地缓存
> 400——客户端请求有错（譬如可以是安全模块拦截）
> 401——请求未经授权
> 403——禁止访问（譬如可以是未登录时禁止）
> 404——资源未找到
> 500——服务器内部错误
> 503——服务不可用

> 1xx——指示信息，表示请求已接收，继续处理
> 2xx——成功，表示请求已被成功接收、理解、接受
> 3xx——重定向，要完成请求必须进行更进一步的操作
> 4xx——客户端错误，请求有语法错误或请求无法实现
> 5xx——服务器端错误，服务器未能实现合法的请求

常用的请求头部（部分）

> Accept: 接收类型，表示浏览器支持的MIME类型（对标服务端返回的Content-Type）
>
> Accept-Encoding：浏览器支持的压缩类型,如gzip等,超出类型不能接收
>
> Content-Type：客户端发送出去实体内容的类型
>
> Cache-Control: 指定请求和响应遵循的缓存机制，如no-cache
>
> If-Modified-Since：对应服务端的Last-Modified，用来匹配看文件是否变动，只能精确到1s之内，http1.0中
>
> Expires：缓存控制，在这个时间内不会请求，直接使用缓存，http1.0，而且是服务端时间
>
> Max-age：代表资源在本地缓存多少秒，有效时间内不会请求，而是使用缓存，http1.1中
>
> If-None-Match：对应服务端的ETag，用来匹配文件内容是否改变（非常精确），http1.1中
>
> Cookie: 有cookie并且同域访问时会自动带上
>
> Connection: 当浏览器与服务器通信时对于长连接如何进行处理,如keep-alive
>
> Host：请求的服务器URL

> Origin：最初的请求是从哪里发起的（只会精确到端口）,Origin比Referer更尊重隐私
>
> Referer：该页面的来源URL(适用于所有类型的请求，会精确到详细页面地址，csrf拦截常用到这个字段)
>
> User-Agent：用户客户端的一些必要信息，如UA头部等

常用的响应头部（部分）

> Access-Control-Allow-Headers: 服务器端允许的请求Headers
>
> Access-Control-Allow-Methods: 服务器端允许的请求方法
>
> Access-Control-Allow-Origin: 服务器端允许的请求Origin头部（譬如为*）
>
> Content-Type：服务端返回的实体内容的类型Date：数据从服务器发送的时间
>
> Cache-Control：告诉浏览器或其他客户，什么环境可以安全的缓存文档
>
> Last-Modified：请求资源的最后修改时间Expires：应该在什么时候认为文档已经过期,从而不再缓存它
>
> Max-age：客户端的本地资源应该缓存多少秒，开启了Cache-Control后有效
>
> ETag：请求变量的实体标签的当前值
>
> Set-Cookie：设置和页面关联的cookie，服务器通过这个头部把cookie传给客户端
>
> Keep-Alive：如果客户端有keep-alive，服务端也会有响应（如timeout=38）
>
> Server：服务器的一些相关信息

解析页面流程

流程简述

浏览器内核拿到内容后，渲染步骤大致可以分为以下几步：

> 1. 解析HTML，构建DOM树
> 2. 解析CSS，生成CSS规则树
> 3. 合并DOM树和CSS规则，生成render树
> 4. 布局render树（Layout/reflow），负责各元素尺寸、位置的计算
> 5. 绘制render树（paint），绘制页面像素信息

> 6. 浏览器会将各层的信息发送给GPU，GPU会将各层合成（composite），显示在屏幕上