咕泡学院 VIP 课：分布式系统的基础 HTTP 协议

课程目标

1. 了解客户端和服务端的请求原理
2. HTTP 协议及其组成
3. Https 交互原理分析

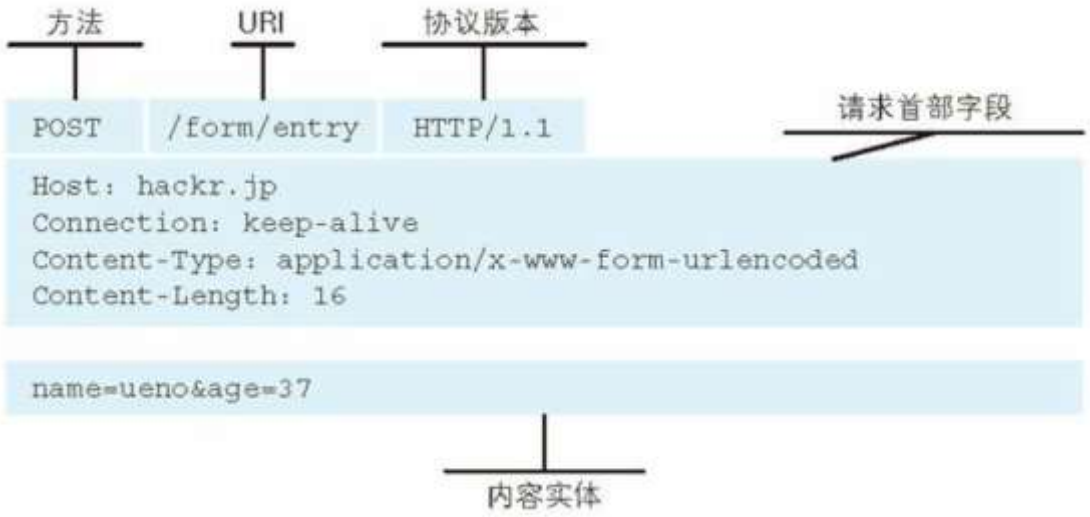
Http 协议的组成

大家可以通过抓包工具，Fillder 或者其他去抓去一个请求，然后可以看到如下的请求数据和响应数据。分为两部分，一个是客户端的请求信息，一个是服务端的响应信息。抓去到的信息如下

request

```
POST https://re.csdn.net/csdnbi HTTP/1.1
方法  url/uri  协议的版本号 1.1
Host: re.csdn.net
Connection: keep-alive
Content-Length: 167
```

```
Accept: */*
Origin: https://www.csdn.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/66.0.3359.23 Safari/537.36
Content-Type: text/plain;charset=UTF-8
Referer: https://www.csdn.net/
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9
Cookie: uuid_tt_dd=10_19119862890-1514946902631-
786149;
__utma=17226283.1502834598.1514952032.1514952032.
1514952032.1;
__utmz=17226283.1514952032.1.1.utmcsr=(direct)|ut
mccn=(direct)|utmcmd=(none); kd_user_id=accb9177-
52d8-41f3-b69e-54bb338ffb23; UN=q331464542;
UM_distinctid=1610314af5bb3a-012f62bad56aa5-
71103742-1fa400-1610314af5ca34;
Hm_ct_6bcd52f51e9b3dce32bec4a3997715ac=1788*1*PC_
VC; BT=1523867282719;
smidV2=20180517165125ad3024b867497a0fbd79f81ef81c
dd4400ceee13dc5e27d30;
dc_session_id=10_1527227855207.688716;
Hm_lvt_6bcd52f51e9b3dce32bec4a3997715ac=152741308
2,1527413263,1527413731,1527415074;
Hm_lpvt_6bcd52f51e9b3dce32bec4a3997715ac=15274229
24; dc_tos=p9dz2k
--------------
[{"headers":{"component":"enterprise","datatype":
"re","version":"v1"},"body":"{\"re\":\"ref=-
&mtp=4&mod=ad_popu_131&con=ad_content_2961%2Cad_o
rder_731&uid=-&ck=-\"}"}]
```



response

```
HTTP/1.1 200 OK
协议版本号 响应状态码 状态码对应的原因
Server: openresty
Date: Sun, 27 May 2018 12:08:44 GMT
Transfer-Encoding: chunked
Connection: keep-alive
Keep-Alive: timeout=20
Access-Control-Allow-Origin: https://www.csdn.net
Access-Control-Allow-Methods: GET, POST, OPTIONS
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: DNT,X-
CustomHeader,Keep-Alive,User-Agent,X-Requested-
With,If-Modified-Since,Cache-Control,Content-
Type,body

2
ok
0
```

URL（Uniform Resource Locator）

URL(Uniform Resource Locator) 地址用于描述一个网络上的资源，基本格式

例 如 ：

http://www.gupaoedu.com:80/java/index.html?name=mic#head

schema://host[:port#]/path/.../?[url-params]#[query-string]

scheme	指定应用层使用的协议(例如：http, https, ftp)
host	HTTP 服务器的 IP 地址或者域名
port#	HTTP 服务器的默认端口是 80，这种情况下端口号可以省略。如果使用了别的端口，必须指明，例如 http://www.gupaoedu.com:8080/
path	访问资源的路径
query-string	查询字符串
#	片段标识符（使用片段标识符通常可标记出已获取资源中的子资源（文档内的某个位置））

URI(Uniform Resource Identifier)

每个 web 服务器资源都有一个名字，这样客户端就可以根据这个名字来找到对应的资源，这个资源称之为（统一资源标识符）

总的来说：URI 是用一个字符串来表示互联网上的某一个资源。而 URL 表示资源的地点（互联网所在的位置）

方法

HTTP 发起的每个请求，都需要告诉告诉服务器要执行什么动作，那么这个动作就是前面报文中看到的【method】。http 协议中提供了多个方法，不同方法的使用场景也也不一样

GET：一般是用于客户端发送一个 URI 地址去获取服务端的资源（一般用于查询操作）

POST：一般用户客户端传输一个实体给到服务端，让服务端去保存（一般用于创建操作）

PUT：向服务器发送数据，一般用于更新数据的操作

HEAD：用于向服务端发起一个查询请求获取 head 信息，比如获取 index.html 的有效性、最近更新时间等。

DELETE：客户端发起一个 Delete 请求要求服务端把某个数据删除（一般用于删除操作）

OPTIONS：查询指定 URI 支持的方法类型（get/post）

http1.1 还支持 trace(追踪路径)和 connect 方法类型

HTTP 协议的特点

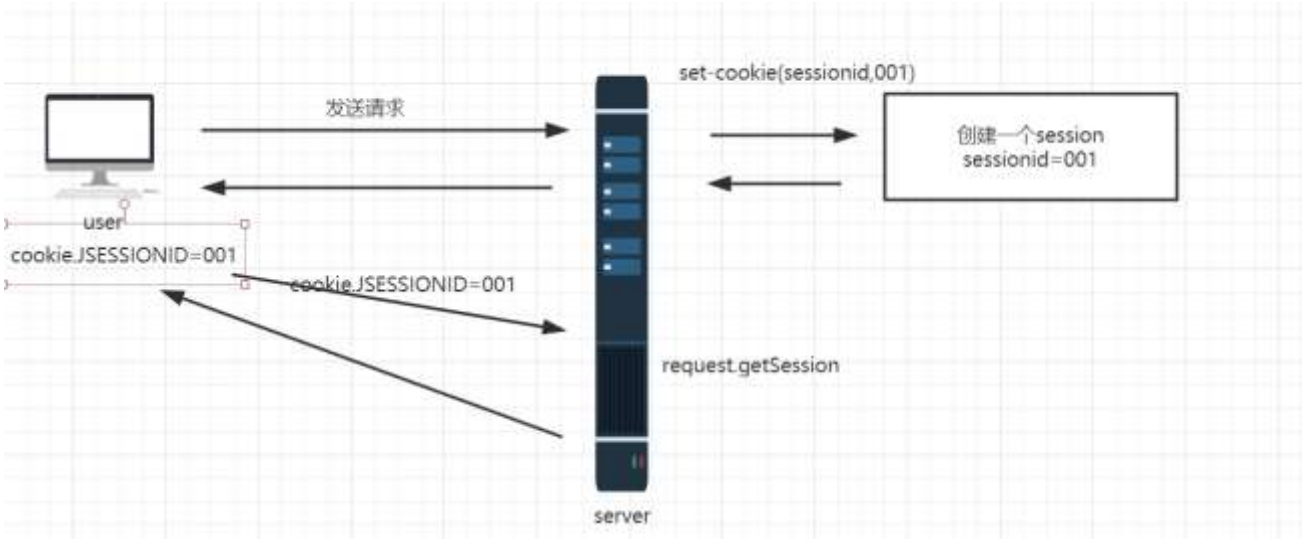
HTTP 协议是无状态的，什么是无状态呢？就是说 HTTP 协议本身不会对请求和响应之间的通信状态做保存。

如何实现有状态的协议

Http 协议中引入了 cookie 技术，用来解决 http 协议无状态的问题。通

过在请求和响应报文中写入 Cookie 信息来控制客户端的状态；Cookie 会根据从服务器端发送的响应报文内的一个叫做 Set-Cookie 的首部字段信息，通知客户端保存 Cookie。当下次客户端再往该服务器发送请求时，客户端会自动在请求报文中加入 Cookie 值后发送出去。

在基于 tomcat 这类的 jsp/servlet 容器中，会提供 session 这样的机制来保存服务端的对象状态。那么整个状态协议的流程就是这样的



HTTP 协议的缺陷

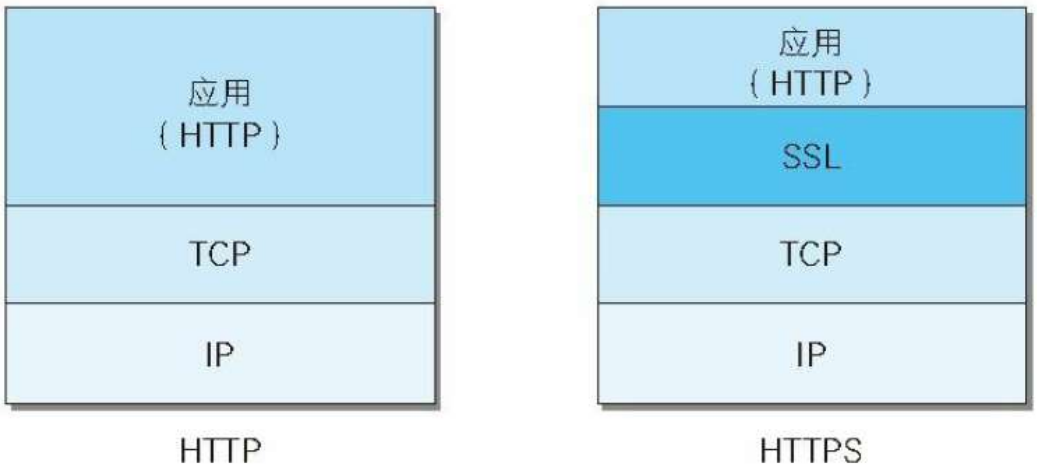
1. 通信过程中是使用明文，内容可能会被窃听
2. 不验证通信双方的身份
3. 无法验证报文的完整性，报文可能被篡改

HTTPS 的原理

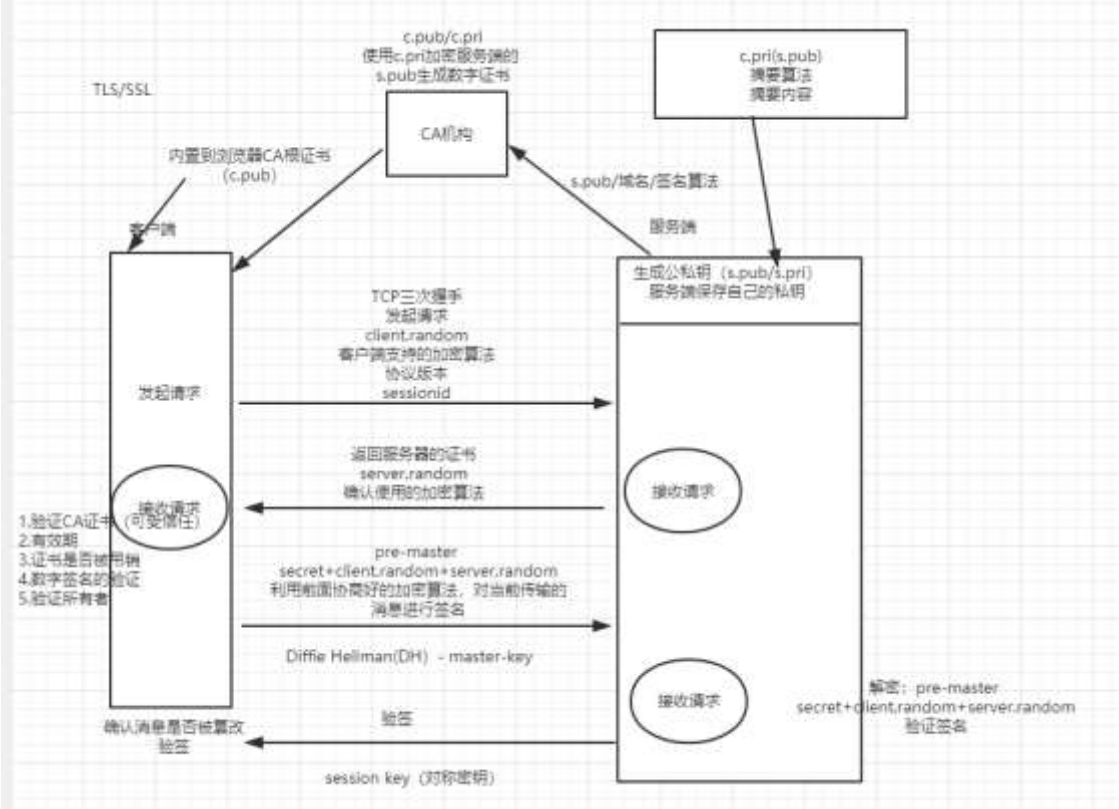
HTTPS 简介

由于 HTTP 协议通信的不安全性，所以人们为了防止信息在传输过程中遭到泄漏或者篡改，就想出来对传输通道进行加密的方式 https。

https 是一种加密的超文本传输协议，它与 HTTP 在协议差异在于对数据传输的过程中，https 对数据做了完全加密。由于 http 协议或者 https 协议都是处于 TCP 传输层之上，同时网络协议又是一个分层的结构，所以在 tcp 协议层之上增加了一层 SSL（Secure Socket Layer，安全层）或者 TLS（Transport Layer Security） 安全层传输协议组合使用用于构造加密通道；



HTTPS 的实现原理



1. 客户端发起请求(Client Hello 包)
   a) 三次握手，建立 TCP 连接
   b) 支持的协议版本(TLS/SSL)
   c) 客户端生成的随机数 client.random，后续用于生成“对话密钥”
   d) 客户端支持的加密算法
   e) sessionid，用于保持同一个会话（如果客户端与服务器费尽周折建立了一个 HTTPS 链接，刚建完就断了，也太可惜）
2. 服务端收到请求，然后响应（Server Hello）
   a) 确认加密通道协议版本
   b) 服务端生成的随机数 server.random，后续用于生成“对话密钥”
   c) 确认使用的加密算法（用于后续的握手消息进行签名防止篡改）

d) 服务器证书（CA 机构颁发给服务端的证书）

3. 客户端收到证书进行验证

a) 验证证书是否是上级 CA 签发的，在验证证书的时候，浏览器会调用系统的证书管理器接口对证书路径中的所有证书一级一级的进行验证，只有路径中所有的证书都是受信的，整个验证的结果才是受信

b) 服务端返回的证书中会包含证书的有效期，可以通过失效日期来验证 证书是否过期

c) 验证证书是否被吊销了

d) 前面我们知道 CA 机构在签发证书的时候，都会使用自己的私钥对证书进行签名

证书里的签名算法字段 sha256RSA 表示 CA 机构使用 sha256 对证书进行摘要，然后使用 RSA 算法对摘要进行私钥签名，而我们也知道 RSA 算法中，使用私钥签名之后，只有公钥才能进行验签。

e) 浏览器使用内置在操作系统上的 CA 机构的公钥对服务器的证书进行验签。确定这个证书是不是由正规的机构颁发。验签之后得知 CA 机构使用 sha256 进行证书摘要，然后客户端再使用 sha256 对证书内容进行一次摘要，如果得到的值和服务端返回的证书验签之后的摘要相同，表示证书没有被修改过

f) 验证通过后，就会显示绿色的安全字样

g) 客户端生成随机数，验证通过之后，客户端会生成一个随机数

pre-master secret，客户端根据之前的：Client.random + sever.random + pre-master 生成对称密钥然后使用证书中的公钥进行加密，同时利用前面协商好的加密算法，将握手消息取 HASH 值，然后用"随机数加密"握手消息+握手消息 HASH 值（签名）"然后传递给服务器端；（*在这里之所以要取握手消息的 HASH 值，主要是把握手消息做一个签名，用于验证握手消息在传输过程中没有被篡改过。*）

4. 服务端接收随机数
   a) 服务端收到客户端的加密数据以后，用自己的私钥对密文进行解密。然后得到 client.random/server.random/pre-master secret.，再用随机数密码 解密 握手消息与 HASH 值，并与传过来的 HASH 值做对比确认是否一致。
   b) 然后用随机密码加密一段握手消息（握手消息＋握手消息的 HASH 值 ）给客户端
5. 客户端接收消息
   a) 客户端用随机数解密并计算握手消息的 HASH，如果与服务端发来的 HASH 一致，此时握手过程结束，
   b) 之后所有的通信数据将由之前交互过程中生成的 pre master secret / client.random/server.random 通过算法得出 session Key，作为后续交互过程中的对称密钥