

# The HTTP Protocol

---

- HTTP
-

Copyright (c) 2013 -2015 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to [youngwlim@hotmail.com](mailto:youngwlim@hotmail.com).

This document was produced by using OpenOffice and Octave.

# HTTP

## Client

`http://host:port/path?query`

### URI

An HTTP request from a client to a server is simply sent using TCP

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: 
Accept: 
Accept-Language: 
Accept-Encoding: 
Accept-Charset: 
Keep-Alive: 
Connection: 
Referer:
```

HTTP Req Header

## Server

### HTML

### HTTP Rsp Header

```
HTTP/1.1 200 OK
Date: 
Server: 
Last-Modified: 
Etag: 
Accept-Ranges: 
Content-Length: 
Connection: 
Content-Type:
```

HTTP Req



HTTP Rsp



# From URI to HTTP Req

http://www.google.com/search?q=Embedded+Web+Server

HTTP Req Header

```
GET /search?q=Embedded+Web+Server HTTP/1.1
Host: www.google.com
User-Agent: 
Accept: 
Accept-Language: 
Accept-Encoding: 
Accept-Charset: 
Keep-Alive: 
Connection: 
Referer:
```

HTTP Req Body (Empty)

HTTP Req Body

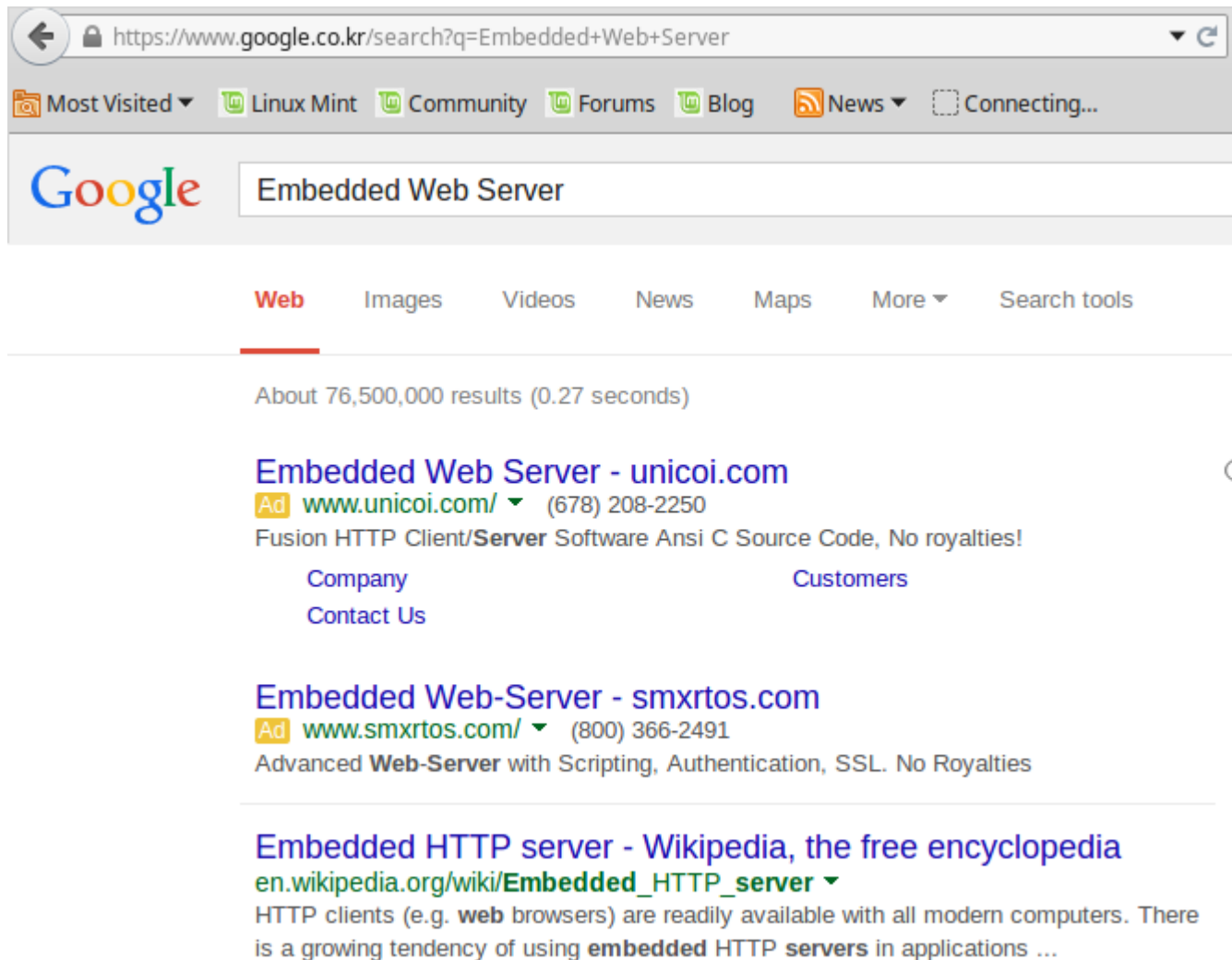
HTTP Req Header

```
POST /search HTTP/1.1
Host: www.google.com
User-Agent: 
Accept: 
Accept-Language: 
Accept-Encoding: 
Accept-Charset: 
Keep-Alive: 
Connection: 
Referer:
```

q=Embedded+Web+Server

HTTP Req Body

# HTTP Rsp Result Display



The screenshot shows a web browser window with the address bar containing the URL `https://www.google.co.kr/search?q=Embedded+Web+Server`. The search bar contains the text "Embedded Web Server". Below the search bar, the "Web" tab is selected. The search results show approximately 76,500,000 results found in 0.27 seconds. The first result is an advertisement for "Embedded Web Server - unicoi.com" with the URL `www.unicoi.com/` and phone number (678) 208-2250. The description reads: "Fusion HTTP Client/Server Software Ansi C Source Code, No royalties!". Below the description are links for "Company", "Customers", and "Contact Us". The second result is an advertisement for "Embedded Web-Server - smxrtos.com" with the URL `www.smxrtos.com/` and phone number (800) 366-2491. The description reads: "Advanced Web-Server with Scripting, Authentication, SSL. No Royalties". The third result is a Wikipedia entry titled "Embedded HTTP server - Wikipedia, the free encyclopedia" with the URL `en.wikipedia.org/wiki/Embedded_HTTP_server`. The snippet reads: "HTTP clients (e.g. web browsers) are readily available with all modern computers. There is a growing tendency of using embedded HTTP servers in applications ...".

<http://www.google.com/search?q=Embedded+Web+Server>

# HTTP Rsp

## HTTP Rsp Header

HTTP/1.1 200 OK

Date: [REDACTED]

Server: [REDACTED]

Last-Modified: [REDACTED]

Etag: [REDACTED]

Accept-Ranges: [REDACTED]

Content-Length: [REDACTED]

Connection: [REDACTED]

Content-Type: [REDACTED]

HTTP Rsp Body  
(HTML contents)

```
<!DOCTYPE html>
<html itemscope="" itemtype="http://schema.org/SearchResultsPage"
lang="en-KR"><head>
<meta http-equiv="content-type" content="text/html;
charset=UTF-8"><meta content="/images/google_favicon_128.png"
itemprop="image"><title>Embedded Web Server - Google Search</title>
<script>(function(){window.google=
{KEI:'HY_uVNHYMMa7mgWR4oLYCg',kEXPI:'3700279,3700362,4011550,4011552,401
{en:0,bv:20,pm:'i',u:'c9c918f0',qbp:0,rre:false},kSID:'HY_uVNHYMMa7mgWR4
KR'};})();(function(){google.lc=[];google.li=0;google.getEI=function(a)
{for(var b;a&&(!a.getAttribute)||!(b=a.getAttribute
("eid")));)a=a.parentNode;return b||google.kEI};google.getLEI=function
(a){for(var b=null;a&&(!a.getAttribute)||!(b=a.getAttribute
("leid")));)a=a.parentNode;return b};google.https=function()
{return"https:"==window.location.protocol};google.ml=function()
{};google.time=function(){return(new Date).getTime
()};google.log=function(a,b,e,f,l){var d=new
Image,h=google.lc,g=google.li,c="",m=google.ls||"";d.onerror=d.onload=d.
(){delete h[g];h[g]=d;if(!e&&-1==b.search("&ei=")){var k=google.getEI
(f),c+"&ei="+k;-1==b.search("&lei=")&&(f=google.getLEI(f))?c
+ "&lei="+f:k!=google.kEI&&(c+ "&lei="+google.kEI)}}
a=e||"/"+(l||"gen_204")+ "?atyp=i&ct="+a+ "&cad="+b+c+m
+ "&zx="+google.time();/^http:\/i.test(a)&&google.https()?(google.ml
(Error("a")).!1.{src:a.almm:1}).delete h[f]):(d.src=a.aoogle.li=a
```

## HTTP Rsp Body

# HTTP Request

---

`http://host:port/path?query`

```
GET /index.html HTTP/1.1
Host: www.example.com
```

GET method

**field: value**

**field:**

Host:

User-Agent:

Refer:

Accept:

Accept-Language:

Accept-Encoding:

Accept-Charset:

# Response (1)

## Status codes

### HTTP/1.1 200 OK

Date: Mon, 23 May 2005 22:38:34 GMT  
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)  
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT  
ETag: "3f80f-1b6-3e1cb03b"  
Accept-Ranges: none  
Content-Length: 438  
Connection: close  
Content-Type: text/html; charset=UTF-8

1xx: Informational  
2xx: Success  
3xx: Redirection  
4xx: Client Error  
5xx: Service Error

1xx: Informational

2xx: Success

200 OK:

206 Partial Content:

3xx: Redirection

301 Moved Permanently

304 Not Modified:

307 Temporary Redirect

4xx: Client Error

400 Bad Request

401 Unauthorized

404 Not Found

406 Not Acceptable

415 Unsupported Media Type

5xx: Service Error

500 Internal Server Error

501 Not Implemented

503 Service Unavailable



# Response (2)

## Field: value

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: none
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
```

```
Date:
Server:
Etag:
Content-Length:
Content-Type:
Content-Encoding:
Location:
```

# HTML Forms

## Form Handling

Client side languages: Javascript

Server side languages: PHP, JSP

### form.html

```
<html>
<body>
  <form action="form_handler.php" method="GET">
    User Name: <input name="user" type="text" />
    <input type="submit" />
  </form>
</body>
</html>
```

### form\_handler.php

```
<html>
<body>
  <?php
    // This will print whatever the user put
    $name = $_GET['user'];
    echo "Hello, ". $name . "!";
  ?>
</body>
</html>
```

# HTTP Request Methods

## Form Handling

Client side languages: Javascript

Server side languages: PHP, JSP

## GET Method

Requests a representation of the specified resource.

Requests using GET should only retrieve data and should have no other effect.

Query string is in the URI

```
GET [redacted] [redacted]
Host: www.example.com
```

## POST Method

The POST request method is designed to request that a web server accepts the data enclosed in the request message's body for storage.

Query string will be placed in the body of the HTTP request

```
POST [redacted] [redacted]
Host: www.example.com
Content-Type: [redacted]
Content-Length: [redacted]
an empty line
[redacted]
```

# URL Encoding

---

## key value pairs

Name: Jonathan Doe

Age: 23

Formula:  $a + b == 13\%$ !

## application/x-www-form-urlencoded

Name=Jonathan+Doe&Age=23&Formula=a+%2B+b+%3D%3D+13%25%21

# GET Method

`http://host:port/path?Name=Jonathan+Doe&Age=23&Formula=a+%2B+b+%3D%3D+13%25%21`

## GET Method

`GET /path?Name=Jonathan+Doe&Age=23&Formula=a+%2B+b+%3D%3D+13%25%21 HTTP/1.1`

the HTTP GET request method is designed to **retrieve** information from the server.

As part of a GET request, **some data can be passed within the URI's query string**, specifying for example search terms, date ranges, or other information that defines the query.

```
GET [redacted] [redacted]
Host: www.example.com
```

# POST Method

```
http://host:port/path?Name=Jonathan+Doe&Age=23&Formula=a+%2B+b+%3D%3D+13%25%21
```

## POST Method

```
POST /path HTTP/1.1
```

```
HOST: ...
```

```
Content-Type: application/X-www-form-urlencoded
```

```
Content-Length: ...
```

```
Empty line
```

```
Name=Jonathan+Doe&Age=23&Formula=a+%2B+b+%3D%3D+13%25%21
```

The POST request method is designed to **request** that a web server **accept** the data enclosed in the request message's body **for storage**. It is often used when **uploading a file** or **submitting a completed web form**.

```
POST [redacted] [redacted]
Host: www.example.com
Content-Type: [redacted]
Content-Length: [redacted]
an empty line
[redacted]
```

# Authentication

---

- IP-address-based
- Form-based
- HTTP Basic
- HTTP Digest

# Sessions

---

Hidden Form Fields  
Cookies



# SSL and TLS

---

Secure Sockets Layer (SSL)  
Transport Layer Security (TLS)

# Client and Server

---

## HTTP architectures

# Proxy

---

Proxy

# Gateway

---

Gateway

# Tunnel

---

Tunnel

# Intermediate Nodes

---

A chain of intermediate systems

# Gnuplot and Canvas

In **Gnuplot**,

file:///home/young/myplot.html

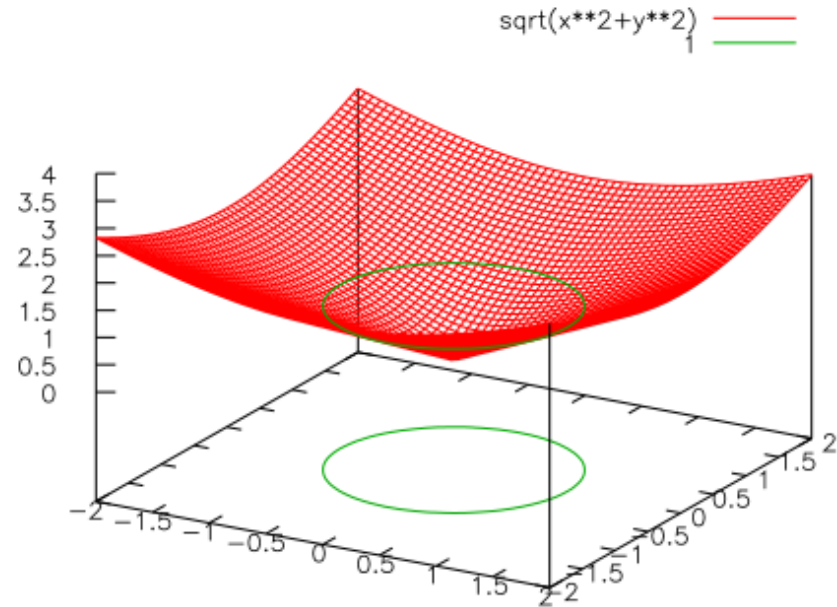
W Wil

```
set terminal 'canvas'  
set output 'myplot.html'
```

```
set view 60, 30, 0.85, 1.1  
set samples 60, 60  
set isosamples 61, 61
```

```
set contour both  
set cntrparam levels discrete  
1, 4
```

```
set xrange [-2: 2]  
set yrange [-2: 2]  
set zrange [0: 4]  
plot sqrt(x**2+y**2)
```



# Reference

---

## References

- [1] <http://en.wikipedia.org/>
- [2] <http://www.w3schools.com/>
- [3] K.H. Koh, HTML, CSS, Javascript (in Korean)
- [4] Gnuplot manual