Web w/ Tutorial

SPARCS WHEEL SEMINAR 2015-07-13 SAMJO

Index

- World Wide Web
  - Uniform Resource Identifier
  - HTTP Request / Response
  - Cookie / Session
- Apache w/ Tutorials
  - Multi-Processing Module
  - Configuration (mods, vhost)
- Nginx w/ Tutorials

With Tutorial

- OS: Debian GNU/Linux 8.0

- **bold orange color** – 이 명령어를 실행시켜주세요!
  - **sudo** 는 생략되어 있습니다.
  - **apt-get** 등의 명령어를 사용할 때에는 **sudo**를 붙여주세요!

Web?

"The **World Wide Web** is an information system of interlinked **hypertext documents** and other digital resources that are accessed via the Internet"

Sir. Tim Berners-Lee (1955-06-08 ~)

History of Web

- 1989: CERN에서 자료공유 목적으로 hypertext라는 개념을 도입
  - “imagine, then, the references in this document all being associated with the network address of the thing to which they referred, so that while reading this document you could skip to them with a click”
- 1990: First web server, web browser (WorldWideWeb), web site (http://line-mode.cern.ch/www/hypertext/WWW/TheProject.html)
- 1991: WWW가 CERN외부에서 최초로 서비스됨

Core of WWW

- Uniform Resource Identifier (URI)
  - 인터넷에 있는 자원을 나타내는 주소
  - http://www.example.com/ab/cd/ee.html?q=asdf#1234
- HyperText Markup Language (HTML)
  - 웹 페이지를 나타내는 형식
- HyperText Transfer Protocol (HTTP)
  - 서버와 클라이언트가 웹 페이지를 주고 받는 형식

Uniform Resource Identifier

- 자원을 식별하는데 사용되어지는 문자열

- URL (~ Locater)와 URN (~ Name)으로 나눠짐

- URL은 웹 자원을 이름으로 식별하는데 사용됨
- URN은 특정 namespace에서 자원을 이름으로 식별하는데 사용
  - ex: ISBN (International Standard Book Number)
  - urn:<namespace id>:<namespace-specific string>
  - ex: urn:isbn:04515450523

Uniform Resource Locator (URL)

scheme://[user:password@]domain:port/path?query#fragment

- scheme: protocol의 이름 (http, https, ftp, file ...)
- domain: 도메인 이름 (naver.com) 또는 IP주소
- port: 포트 번호 (생략시 http=80, https=443, ftp=21 ...)
- path: 자료 경로 (/sambradjo/post/907269945997390)
- query: 웹 서버에 보내는 추가적인 정보
  - ?name1=data1&name2=data2 ...
- fragment: 문서의 특정 위치나 부분을 지칭

URL Examples

- http://www.naver.com/
- http://sambradjo.net:8080/
- ftp://samjo:samsam@sambradjo.net/file1.pptx
- mailto:samjo@sparcs.org
- file:///home/samjo/abc.pdf
- http://sambradjo.net/login/?id=samjo&pw=samsam
- http://sambradjo.net/web/url.html#history

URL - Extra Info

- 사용 가능한 문자열: ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789 - ~ _ .
  - path, query에 다른 문자열이 있을 경우 escape해야 함
  - domain에서는 ~ _ . 사용 불가

- case-insensitive: scheme, domain
- case-sensitive: **path**, query, fragment

- Internationalized domain name (IDN)
  - Unicode 도메인 이름을 지원
  - http://xn--220b31d95hq8o.xn--3e0b707e/

HyperText Markup Language (HTML)

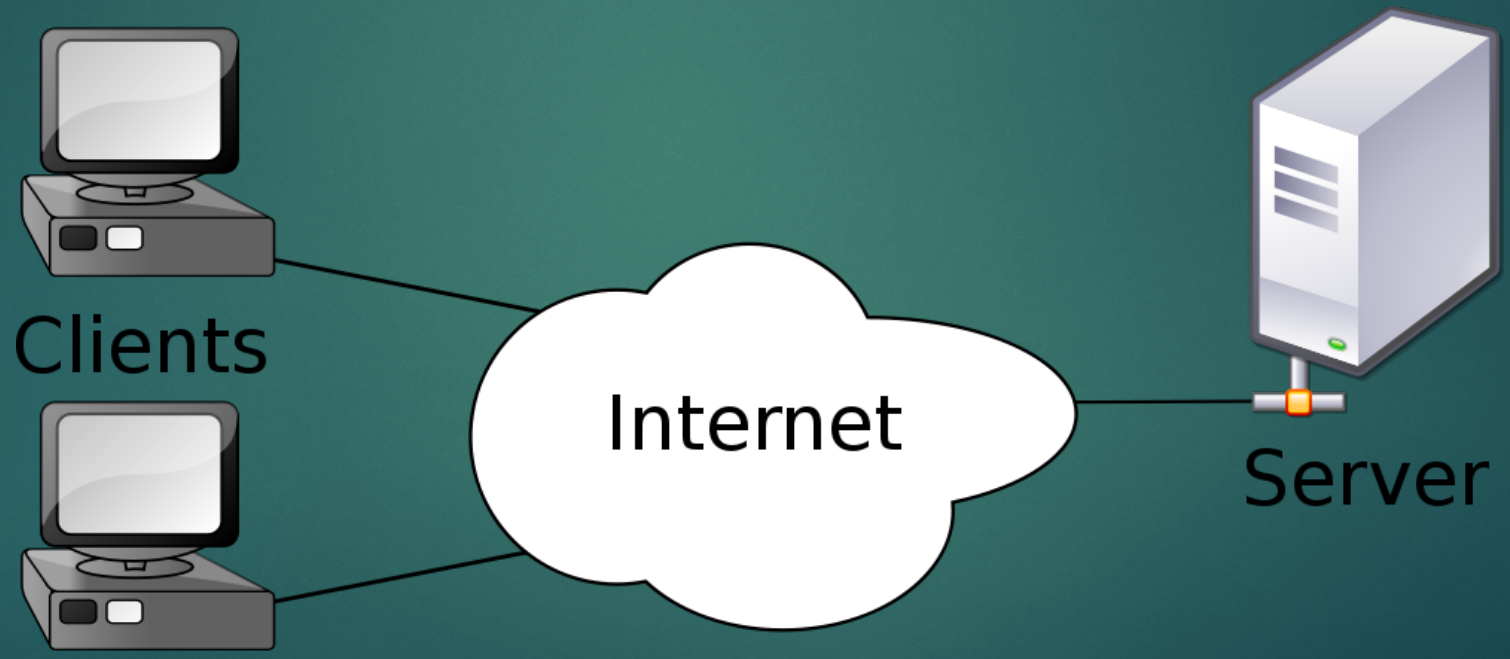
- “The standard markup language used to create web pages”
- Developed by W3C & WHATWG
  - World Wide Web Consortium
  - Web Hypertext Application Technology Working Group
- 기본적인 구조에 대해서는 다들 아시니 PASS
  - <abc attr1=“value1”></abc>
  - <abc attr1=“value1”/>

Brief History of HTML

- HTML 1.0 (1991) -> HTML 2.0 (1994) -> HTML 3.2 (1997)
  - Age of Mosaic, Netscape, IE 3.0
- HTML 4.0 (1997) -> HTML 4.01, XHTML 1.0 (2000)
  - Age of IE 4.0 ~ 6.0
- HTML 5.0 (2014)
  - Chrome, Firefox, Internet Explorer, Safari, Opera

HyperText Transfer Protocol

- "an **application protocol** for distributed, collaborative, hypermedia information systems"



HTTP Basic

- Located in OSI Layer 7, Port: TCP 80

- Standards
  - HTTP/1.0 (1996)
  - HTTP/1.1 (1999): improvement in caching, connection optimization…
  - HTTP/2 (2015): compression on http headers, …

- Client -> Server: Request
- Server -> Client: Response

HTTP Request Structure

[method] [URI] [Version]

[header field]:[header value]*

CRLF [body]

GET / HTTP/1.1

Host: www.naver.com

Connection: keep-alive

http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html

HTTP Request Method

- **GET**: URL에 해당하는 자료의 요청; 대부분의 페이지 이동
  - Query는 URL 끝에 붙음 (ex: abc.com/?id=sam&pw=abc
- **POST**: 서버에 자료를 보내는 요청; 대부분의 버튼 클릭
  - Query는 body에 삽입되어 전송됨
- PUT: 해당 URL에 자료를 저장
- DELETE: 해당 URL에 자료를 삭제
- TRACE, OPTIONS, CONNECT, HEAD

HTTP Response Structure

[version] [status code] [reason message]

[header field]:[header value]*

CRLF [body]

HTTP/1.1 200 OK

Content-Type: text/html; charset=UTF-8

CRLF

<!doctype html> ...

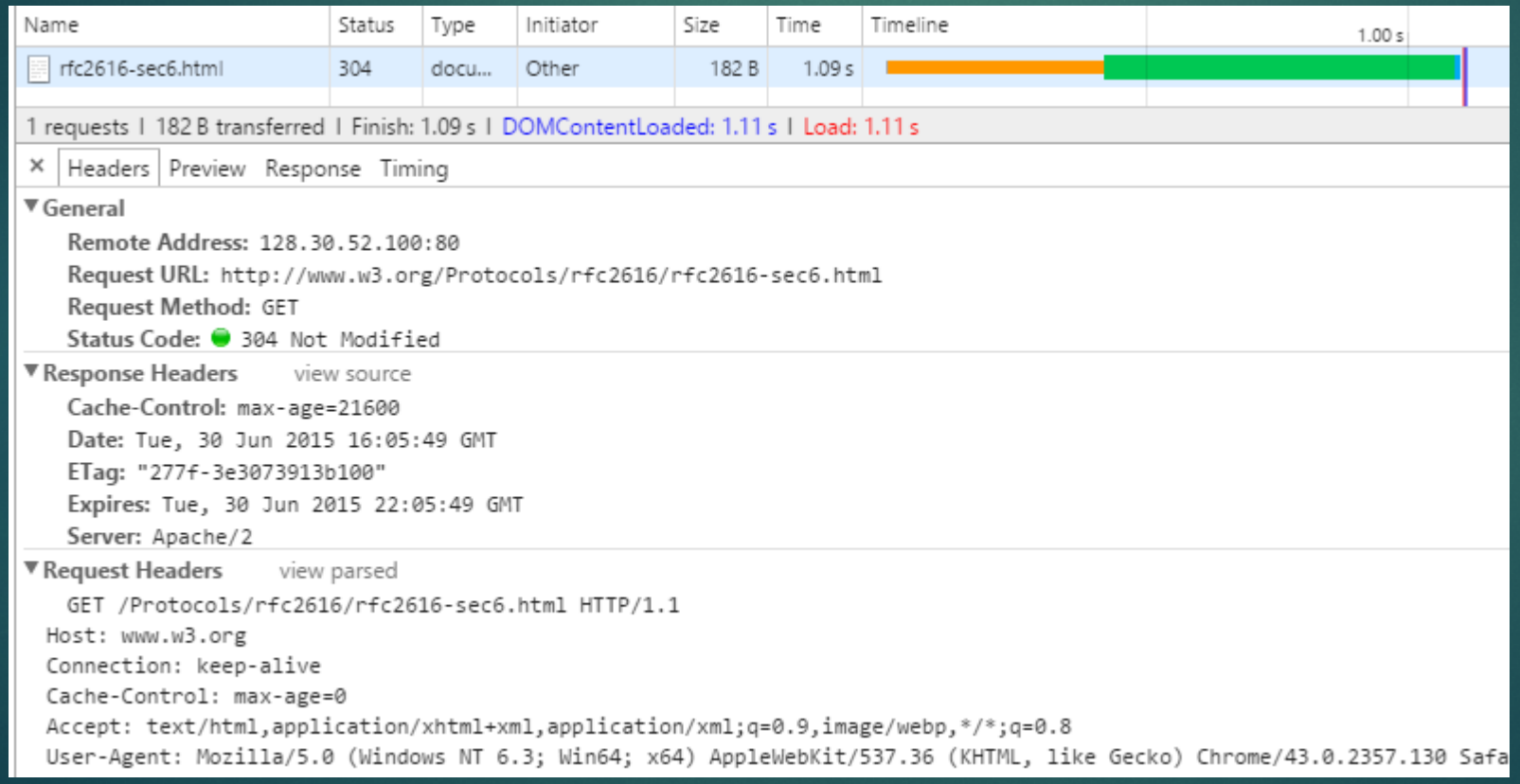
http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html

HTTP Status Code

- 1xx: 조건부 응답
- 2xx: 성공
  - 200: 성공
- 3xx: 리다이렉션 완료
  - 301: 영구 이동; 302: 임시 이동
  - 304: 수정되지 않음
- 4xx: 요청 오류
  - 400: 잘못된 요청; 401: 권한 없음; 403: 금지됨; 404: 찾을 수 없음
- 5xx: 서버 오류
  - 500: 내부 오류; 503: 서비스 사용 불가

HTTP Packet

- Chrome에서 F12를 눌러 개발자 도구를 연다
- Network탭을 열고, 웹 페이지를 하나 로드한다.



User-Agent

- Web Browser의 종류를 서버에게 알려주기 위해 만든 header
- 서버는 이 문자열로 사용자의 web browser을 식별할 수 있음
  - **주의: 매우 당연하게 이를 다른 값으로 바꿔서 보낼 수 있음**
- ex: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.130 Safari/537.36

Cookie

- “a small piece of data sent from a website and **stored in a user’s web browser** while the user is browsing that website”
  - 웹 사이트가 사용자의 정보를 기억하는데 사용
  - 로그인, 온라인 쇼핑몰 쇼핑 카트, 광고 등에 사용
  - **사용자의 웹 브라우저에 저장되므로 값을 믿어서는 안됨**

- 1. 웹 사이트가 Response Header의 Set-Cookie라는 항목에 쿠키 정보를 만들어 보냄
- 2. 웹 브라우저는 이를 사용자 컴퓨터에 저장함
- 3. 해당 사이트를 방문할 때마다 Request Header에 넣어서 보냄

Session

- 웹 서버 쪽에서 사용자에 대한 정보를 저장함
  - 웹 서버가 해킹 당하지 않는 이상 cookie와 달리 데이터를 신뢰할 수 있음
  - 저장 데이터 형식 및 크기에 제한이 없음 (cookie의 경우 4KB로 제한)

- 1. 처음 요청시 고유한 id를 만들어 사용자에게 cookie로 전송
- 2. 사용자에게 id를 받아서 해당 id에 맞는 정보를 서버쪽에서 찾음

- ex: PHP의 경우 처음 사이트 접속시 PHPSESSID라는 고유한 id를 만들고 이 cookie를 사용자에게 전달함. 이후 사용자가 해당 웹 사이트에 요청을 보내면, 이 id로 사용자를 식별하여 서버쪽에 저장된 데이터를 불러옴

Core of WWW

- 1. Content: .html, .txt, .jpg, .avi
- 2. User: 나, 너 그리고 우리
- 3. Client: web browser (ex: Chrome, IE, ...)
- 4. Server: ...?????
  - Apache
  - Nginx
  - ~~IIS (Internet Information Server)~~

Apache

- 세계에서 가장 많이 쓰는 웹 서버 SW
  - 54.2% of all active website (2013/06)
- Apache 재단에서 개발
- 최신 버전: 2.4.12
- Cross-Platform
  - Linux, Mac, Windows
  - FreeBSD, Solaris, OS/2 …

- 세계에서 가장 유명한 공격 헬기
  - 약 2000대 생산(2013/03)
- 보잉사에서 개발
- 최신 기종: AH-64F
- 다양한 국가에서 운용
  - U.S, U.K, Israel
  - Netherlands, Egypt …

Version

- Apache 1.3 (1998-06-06 ~ 2010-02-03)
- Apache 2.0 (2002-04-06 ~ 2013-07-10)
- Apache 2.2 (2015-12-01 ~ 2014-08-03)
- Apache 2.4 (2012-02-21 ~ 2015-01-29)

- **주의: Apache 2.4에서는 Apache 2.2에서 쓰던 설정 syntax가 동작하지 않을 수 있다!**
  - **http://httpd.apache.org/docs/2.4/en/upgrading.html**

- **이 세미나에서는 Apache 2.4를 기준으로 설명합니다!**

Apache Overview

- **apt-get install apache2**
- **service apache2 [start | stop | restart | reload | status]**
  - 설정 파일을 바꾸고 나서는 반드시 restart 또는 reload를 해주자!

- /etc/apache2: root directory
  - Apache2의 설정 파일 및 바이너리 파일
- /var/www: site root
  - 이 폴더 안의 디렉터리 구조 ➔ URL의 path
- /var/log/apache2: log files

Apache Conf Files / Mods

- /etc/apache2/apache2.conf: 기본 설정
- /etc/apache2/ports.conf: 포트 설정
- /etc/apache2/conf.d/: 추가적인 설정 파일들
  - ex: php5.conf – PHP 설치시 생성; PHP 관련 설정을 저장함
- /etc/apache2/sites-available: 사용 가능한 vhost
- /etc/apache2/sites-enabled: 활성화된 vhost
- /etc/apache2/mods-available: 사용 가능한 mods
- /etc/apache2/mods-enabled: 활성화된 mods

- ...???????????

Apache MPM

- 하나의 apache 프로세스가 모든 요청을 처리한다면
  - 동시에 많은 사용자가 접속할 때 속도가 매우 느려짐
  - 하나의 요청에 문제가 생겨서 프로세스가 터지면, 모든 요청을 다 받을 수 없음
- Multi-Processing Module: Apache 메인 프로세스가 요청을 자식 프로세스에게 분배하여 처리하도록 하는 모듈
- 세 가지 방법: Prefork, Worker, Event

Apache MPM - Prefork

- 한 개의 process가 한 개의 thread
- 한 개의 thread가 한 개의 연결을 담당

- 10000명이 접속하면 10000개의 process
  - 메모리를 많이 잡아먹음

- 1개의 process가 터지면 1명의 연결만 끊김
  - 안정적인 서비스 가능

- Single or double core에서 선호

Apache MPM - Worker

- 한 개의 process가 여러개의 thread
- 한 개의 thread가 한 개의 연결을 담당

- Prefork 방식보다는 메모리를 덜 사용함

- 1개의 process가 터지면 여러명의 연결이 끊길 수 있음
  - Prefork보다는 상대적으로 불안함

- Multi core에서 선호

Apache MPM - Event

- Prefork, Worker: “한 개의 thread가 한 개의 연결을 담당”
  - ⇒ 연결이 살아있으면(KeepAlive), 다음 요청이 들어올 때까지 계속 thread가 대기 상태로 있어야 함
- Event: 요청을 받는 thread / 요청을 처리하는 thread를 따로 분리
- Apache 2.4부터 stable한 기능으로 추가

Keep Alive?

- 원래의 HTTP라면 서버는 클라이언트가 보낸 요청에 대해 적절한 응답을 하고, 연결을 종료함

- KeepAlive를 on으로 해놓으면 KeepAliveTimeOut 동안 접속을 끊지 않고 다음 요청을 기다림
  - 2개의 process, 2명의 사용자가 각각 100번의 요청을 보낸다면...?

- 무조건 on으로 설정할 것이 아니라, 접속자 수 및 메모리 용량에 따라 결정해야 함
  - 동접이 10000명인 사이트라면, 10000개의 thread!

Apache2.conf - MPM

<IfModule mpm_[prefork | worker | event]_module>

StartServers: 시작시 process의 갯수

ServerLimit: process의 최대 개수

MaxRequestWorkers: 동시에 사용가능한 process / thread의 개수

MinSpareThreads: 놀고 있는 thread의 최소 개수

MaxSpareThreads: 놀고 있는 thread의 최대 개수

ThreadLimit: process가 가지는 최대 thread 개수

ThreadsPerChild: process가 계속 가지는 thread 개수

MaxConnectionsPerChild: process / thread당 동시 처리 가능한 최대 요청

</IfModule>

Apache Mods

- = Modules = Plug-ins = Addon

- core, mpm_[common | prepork | worker | event]
  - 아파치 기본 및 다중처리 관련 모듈
- alias, rewrite: URL 리다이렉션
- auth_basic, auth_digest, authn_alias, ...
  - 인증 관련 모듈 (사용자 인증 / 권한 부여 ...)
- cgi: common gate interface 모듈
- cache, ssl, ldap ... more and more ...
- + third-party module!

Common Gateway Interface

- “a standard way for web servers to **interface with executable programs** installed on the server that generate web page dynamically”

- Django: 웹 페이지를 동적으로 생성하기 위한 프레임워크
  - execute: python manage.py ...

- Chrome <-> Apache2: HyperText Transfer Protocol
- Apache2 <-> Django: Common Gateway Interface

- WSGI (Web Server Gateway Interface): CGI for Python

Virtual Host

- 한 서버에서 여러 개의 웹사이트를 운영해보자!

- 다양한 상황
  - IP주소는 하나인데 도메인 이름은 여러 개
  - IP주소와 도메인 이름이 여러 개
  - 다른 IP주소인데 같은 내용을 서비스
  - 여러 포트에서 서로 다른 사이트
    - 80에서는 ara, 8080에서는 OTL

Available / Enabled

- Available: 사용 가능한 것들의 “실제” 파일이 저장되는 곳
- Enabled: 사용할 것들의 symbolic link를 모아놓은 곳
  - apache 실행시 이 곳에 있는 mods / vhost가 로드

- a2enmod / a2dismod <module-name>
- a2ensite / a2dissite <vhost-name>

Apache2.conf

- ServerRoot: apache가 설치된 폴더
- PidFile: Pid 파일을 저장하는 경로; apache root 프로세스가 시작할 때 자신의 pid를 기록
- KeepAlive: On | Off; Listen: 요청을 받을 포트 번호를 기록
- TimeOut: 요청을 실패하기 전까지 기다리는 시간; 60으로 설정해 놓았다면, 요청의 처음 패킷이 들어온때부터 60초 안에 모든 패킷을 받고 / 패킷을 처리하고 (DB 읽기 등) / 패킷을 써야 한다.
- LoadModule: 해당 모듈을 로드한다; Include: 해당 설정 파일을 로드한다
- IfModule: 해당 모듈이 로드되었다면, 블록 안의 내용을 포함한다
- DocumentRoot: 웹에서 접근 가능한 메인 파일 트리를 지정한다
  - ex: DocumentRoot “/data/web/samjo”일 경우

    sambradjo.net/sparcs/abc.html => /data/web/samjo/sparcs/abc.html

Apache2.conf

- Directory: 특정 directory에만 적용하는 설정들을 묶어준다.
  - <Directory /data/web/samjo/abc> ... </Directory>
- Options: 특정 directory에 어떠한 옵션들이 가능한지 설정한다.
  - All, ExecCGI, FollowSymLinks, Includes, Indexes, MultiViews
  - ex: Options +FollowSymLinks -Indexes
- DirectoryIndex: directory를 요청할 때, 찾아볼 파일 목록을 설정한다.
  - ex: DirectoryIndex index.html

    sambradjo.net/abc/ => /data/web/samjo/abc/index.html
- Alias: DocumentRoot가 아닌 다른 폴더를 문서 계층에 포함한다.
  - ex: Alias /media/ /data/samjo/static/

    sambradjo.net/media/abc/a.js => /data/samjo/static/adc/a.js
- AllowOverride: .htaccess파일로 설정을 override 할 수 있는지 설정한다.

.htaccess

- 디렉터리 별로 다른 설정을 적용해보자!
  - ex: /var/www/abc 폴더에는 사용자 인증이 필요, 다른 곳에는 불필요

- 해당 폴더에 .htaccess라는 파일을 만들어 그 안에 내용을 넣으면 됨
- 반드시 apache2.conf에 AllowOverride가 설정되어 있어야 함
- AllowOverride를 설정할 경우 속도가 느려질 수 있음

- http://httpd.apache.org/docs/2.4/ko/howto/htaccess.html

DIY

- Do It Yourself

- Apache는 설정에 관련된 수많은 키워드를 정의하고 있음
- 또한 상황에 따라 많은 경우가 있어 세미나에서 다양한 케이스를 모두 다룰 수 없음
- 그때그때 앞에서 배운 키워드를 기반으로 구글링

- http://httpd.apache.org/docs/2.4/ko/vhosts/
- http://www.stackoverflow.com/

Tutorial

- Tutorial 목표:
  - 1. wseminar*.sparcs.org으로 들어가면 본인이 원하는 html을 띄워보자!
  - 2. wseminar*.sparcs.org:8080으로 들어가면 1과 다른 html을 띄워보자!

- Tutorial 과정:
  - 1. /var/www/html/ 에 index.html 파일을 추가하고, 원하는 내용을 작성한다.
  - 2. ports.conf에 8080 포트를 추가한다.
  - 3. /var/www/html2/ 에 index.html 파일을 추가하고, 원하는 내용을 작성한다.
  - 4. site-available 안에 새 .conf 파일을 만들고 이를 활성화시킨다.

Tutorial Process

- **vi /var/www/html/index.html**
- **vi /etc/apache2/ports.conf**
  - Listen 8080을 추가한다.
- **mkdir /var/www/html2; vi /var/www/html2/index.html**
- **cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/001-new.conf**
- **vi /etc/apache2/sites-available/001-new.conf**
  - <VirtualHost *:80> => <VirtualHost *:8080>
  - /var/www/html => /var/www/html2
- **a2ensite 001-new; service apache2 reload**

NGINX

- 세계에서 두 번째로 많이 쓰이는 웹 서버 SW
  - 14.24% of all active website (2015/04)
- Nginx, Inc.에 의해 개발 (2011년 설립)
- 최신 버전: 1.8.0
- Cross-Platform
  - Linux, Mac, Windows, Unix, Solaris, BSDs …

Apache vs Nginx

- Process, Thread, Event-based vs Event-based
  - Nginx가 concurrent request시 더 적은 메모리를 사용
- Administrative options, .htaccess vs None
  - Nginx는 초기 설정하기 매우 빠르고 편리
  - 그러나 인증이나 각 사용자에 대한 설정은 Apache보다 부족
- “Apache는 무겁지만, 많은 기능과 확장 프로그램이 존재하고, Nginx는 가볍고 빠르지만 상대적으로 기능이 적다.”

Nginx Overview

- **apt-get install nginx** (1.6.2를 사용합니다.)
- **service nginx [start | stop | restart | reload | status]**
  - 설정 파일을 바꾸고 나서는 반드시 restart 또는 reload를 해주자!

- /etc/nginx: root directory
  - Nginx의 설정 파일 및 바이너리 파일

- /var/www: site root
- /var/log/nginx: log files

Nginx Conf Files / Mods

- /etc/nginx/nginx.conf: 기본 설정
- /etc/nginx/conf.d/: 추가적인 설정 파일들
- /etc/nginx/sites-available: 사용 가능한 vhost
- /etc/nginx/sites-enabled: 활성화된 vhost

- mod: ./configure –add-module=/path/to/module/source
- sites: a2ensite / a2dissite처럼 멋진 명령어 따위는 없다.
  - 그냥 ln -s를 사용하여 symbolic link를 걸어주면 된다.

Nginx Settings

- Apache와 syntax상의 차이가 있음
  - <Keyword> ... </Keyword> vs keyword { ... }
  - PascalCase vs lowercase_with_underscore

- Apache와 유사한 키워드가 많고, 좀 더 직관적이므로 나머지 설명은 생략하고 실습때 필요한 것을 설명
  - http://nginx.org/en/docs/

Tutorial

- Tutorial 목표: nginx와 uwsgi를 사용하여 django 프로젝트를 띄워 보자!

- Tutorial 과정:
  - 0. apache2 중지시키기 (**service apache2 stop**)
  - 1. python 및 Django, uwsgi 설치
    - **apt-get install python python-pip python-dev**
    - **pip install django uwsgi**
  - 2. django로 샘플 프로젝트 만들기
    - **mkdir ~/django/; cd ~/django/; django-admin.py startproject sample**
  - 3. nginx 설정하고 uwsgi로 실행하기

Tutorial Process

- **wget https://raw.githubusercontent.com/nginx/nginx/master/conf/uwsgi_params -O sample/uwsgi_params**
- **vi /etc/nginx/sites-available/default; service nginx restart**

```
upstream django { server 127.0.0.1:22223; }
server {
        listen 80; charset utf-8;  server_name wseminar*.sparcs.org;
        location / {
                uwsgi_pass django;
                include /home/samjo/django/sample/uwsgi_params;
        }
}
```

- **cd sample; uwsgi --socket :22223 --wsgi-file sample/wsgi.py**

Q&A

END OF THIS SEMINAR

Supplementary Slide

딱히 WWW / WEB SERVER랑은 상관이 없지만 설명해야 할 것들

Process and Thread

- Process: instance of a computer program that is being executed
  - 자신만의 memory 공간을 가짐 (실행 코드, 관련 데이터 포함)
  - 동시에 코드가 실행되도록 여러 개의 thread를 가질 수 있음

- Thread: smallest sequence of programmed instructions that can be managed **independently** by a scheduler
  - Thread는 process의 구성 요소
  - 같은 process의 thread는 메모리를 공유하고 있음

Symbolic Link

- Special file that contains a reference to another file or dir.
- 절대 경로 및 상대 경로로 참조 가능
- 특별한 처리 없이도 자동으로 OS가 해당 파일 또는 디렉터리를 가져옴
- 원본 파일이 옮겨지거나 삭제된 경우에도 symbolic link는 변하지 않음
- **ln -s target_path link_path**

- c.f Shortcut (바로가기) in Microsoft Windows
  - 해당 파일이 옮겨진 경우 자동으로 같이 변경됨
  - 프로그램에서 특별한 처리를 해줘야 해당 파일 또는 디렉터리를 가져올 수 있음

References

- http://en.wikipedia.org/
- http://nginx.org/en/docs/
- http://httpd.apache.org/docs/
- http://sparcs.org/seminar/
  - 2014 Wheel Seminar – Web Server by protos
  - 2013 Wheel Seminar – 8. Apache by yasik
  - 2012 Wheel Seminar – 12. Web by yasik
  - 2012 Wheel Seminar – 14. Apache by aon