숭실대학교 Soongsil University

Web Programming 기초 II

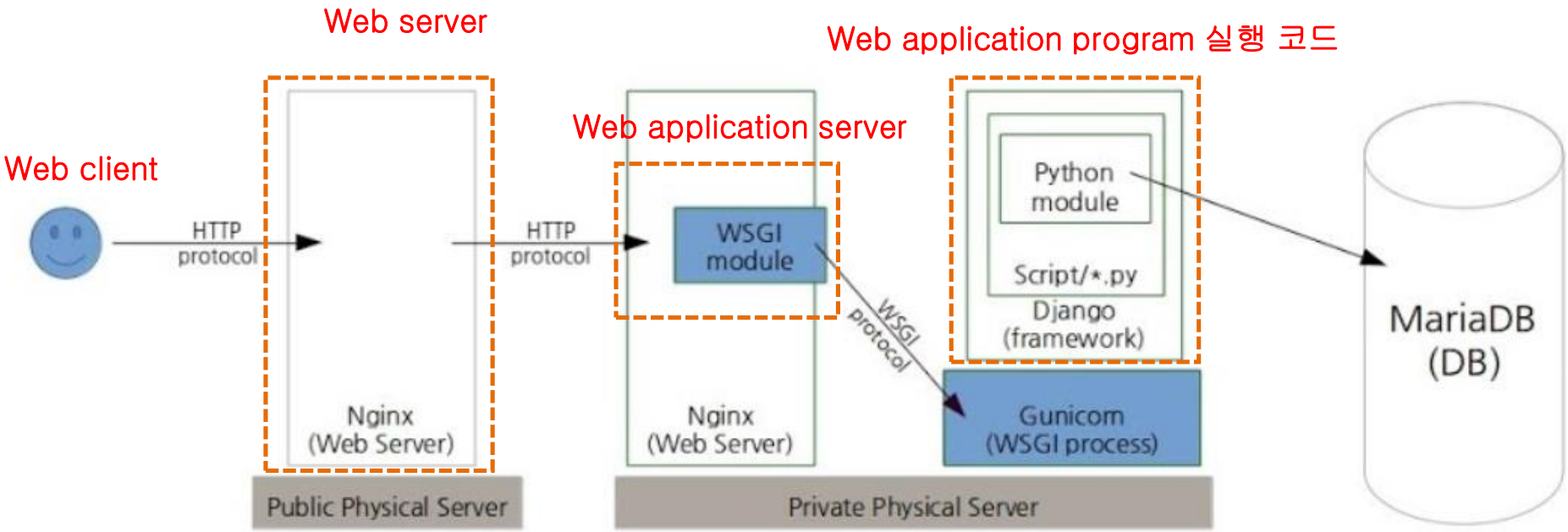
숭실대학교 스마트시스템소프트웨어 학과
Department of Smart Systems Software

Contents

1. Web Server 개요
2. Web Server (Apache, Nginx) 기능
3. Python Web Standard Library
4. Web server program 기초
5. urllib.parse 모듈

Web Server 개요

- Web Client (Web Browser)에서 요청을 받아 처리하는 back-end web application 처리 개념도



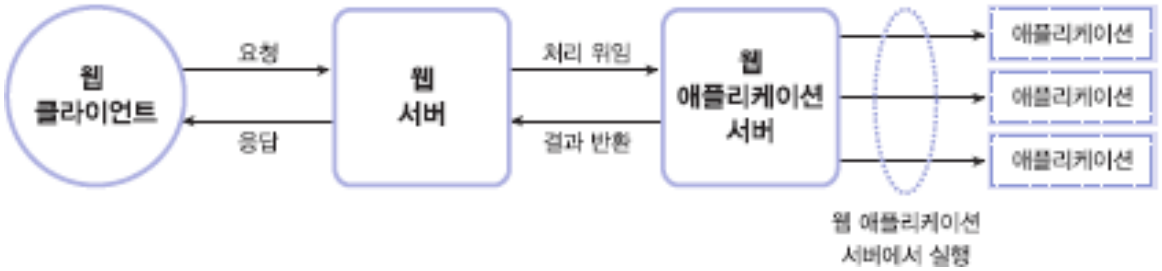
Web Server 개요

- Web server
  - Web browser(client)의 요청을 받아서 처리하고, 그 결과를 web client에게 응답
  - 주로 정적 페이지인 HTML, 이미지, CSS, JavaScript 파일을 web client에게 제공할 때 사용
    - 정적(static) 페이지 예 : 사용자가 web page를 요청할 때 마다 항상 같은 내용을 제공하는 web page (예 : 회사 소개)
  - 동적 페이지(dynamic web page)가 필요하다면, web application server에 처리를 요청하고, web application server가 처리된 결과를 받아 web client에게 넘김
    - 동적 페이지 예 : 게시글 list web page
      - python등의 프로그램에서 DB의 게시 table에 있는 게시글 data를 읽어 html 문서 형태로 변환 처리
  - 제품 예 : Apache httpd, Nginx, lighthttpd, IIS(Microsoft) 등

Web Server 개요

- Web application server
  - Web server로 부터 동적 페이지 요청을 받아 application을 수행하는 실행 코드(예: 게시글 처리 프로그램)를 실행하여, 실행 결과를 web server로 반환
  - DB와 연동이 필요한 Web application 실행 코드를 실행할 때 DB와 연동할 수 있는 기능 제공
  - 제품 예 : Apache Tomcat, JBOSS, WebLogic, WebSphere, Jetty, Jeus, WSGI module, Gunicorn 등
  - WSGI (Web Server Gateway Interface) module : Python 전용 web application server로, Flask, Django등의 web application framework에서 wsgi 표준 모듈 구현되어 있음
- Web application 실행 코드
  - 게시글 처리 프로그램과 같이 python, java등으로 coding한 프로그램을 말하며, Web application server로 부터 호출됨
  - DB 연동을 위한 library를 호출하여 DB table에 read, write 수행

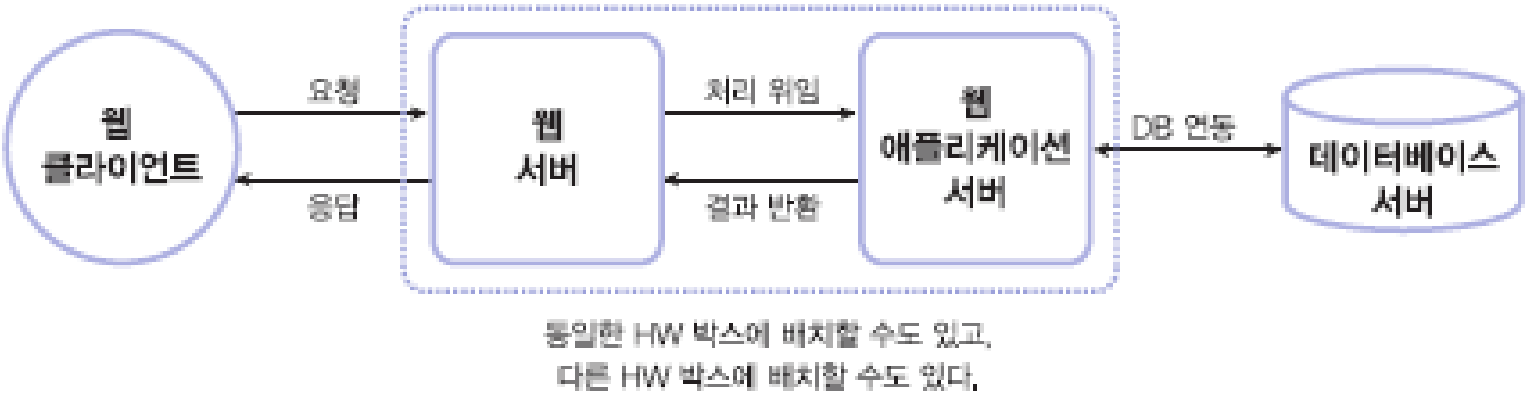
- Web server와 Web application server 분리의 장점
  - Web application server에서 여러 개의 web application 실행 프로그램을 동시 처리하여 처리 속도 개선
    - Web application server에서 동적 페이지를 처리할 경우 정적 페이지를 처리하는 것보다 수 배에서 수십 배의 메모리 소비하여 정적 페이지 대비 처리 속도가 늦어질 수 있음



  - Web application server에서 DB 연동 기능 제공

Web Server 개요

- Web server와 Web application server의 HW 배치
  - 통상 web server, web application server는 software관점의 서버 프로그램을 의미함
  - Web server program과 web application server program을 같은 HW에 설치할 수도 있고, 다른 HW로 분리하여 설치 가능



Web Server 기능

- Static content and more
  - Web server의 원래 목적은 documents를 보여주는 것이지 application 지원 목적이 아님
  - 현재도, web server의 가장 큰 서비스는 static contents(images, HTML files, videos, downloadable files, other media stored on the disk)처리를 지원함
  - Apache, Nginx, IIS등 web server 전용 제품들은 disk storage에 저장된 static contents들을 매우 빠르고 효율적으로 보여주는 기능을 제공함
  - Authenticated users만 특정 static content를 download할 수 있도록 access control하는 기능 제공함
- Routing and load balancing
  - Request routing : web server가 web browser의 개별 request의 내용을 파악하여, 각 request를 처리할 수 있는 web application server로 전달하는 것을 말함

Web Server 기능

- Load balancing : backend web application의 web browse의 request 동시 처리 속도를 높이기 위하여 web application을 여러 server에 복사하여 분산시키는 기법

Concurrent users

- Python web server에서 multi threading 기능을 구현하여 concurrency를 제공한다 하더라도 동시에 수많은 web browser의 request를 처리하기 위해서는 web server 자체가 multi threading을 지원함
- Web server마다 동시에 처리할 수 있는 user 수를 의미함

Caching

- Server side caching : web server가 multi web browser로 부터 같은 request를 받을 경우 첫 번째 request에 대하여 backend web application에서 처리하지만, 두 번째 request에 대해서는 backend web application을 부르지 않고 첫 번째 처리결과가 저장된 cache(temporary storage)내용을 web browser에게 전달함

Web Server 기능

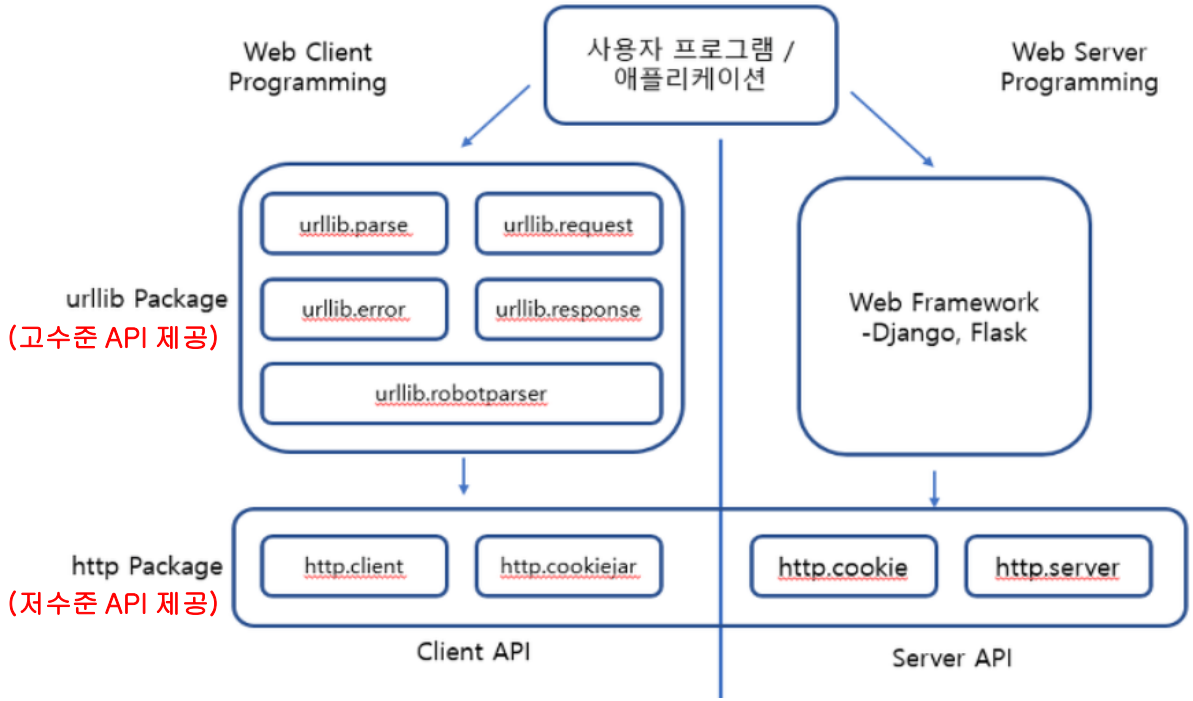
- Web browser cache : browser에서 최근에 전달받은 image등을 cacheable resources에 저장하여, 두 번째 같은 web page를 web server에 요청할 때 web browser에 최근 저장된 cache 내용을 보여주어 처리 속도 개선함

Capacity

- Web serve가 web browser의 multi request에 대한 처리속도를 말함
- Web server capacity를 개선하기 위하여 cache등의 기능을 활용할 수 있음

Python Web Standard Library

- urllib package
  - web client program coding하는데 사용되는 모듈들로 구성
  - urllib.parse 모듈은 web server program에서도 사용됨
- http package : server용과 client용으로 나뉘어 있음
  - Django, Flask는 http.server 등 표준 library 이용하여 구현됨



Python Web Standard Library

- Python 3.x와 2.x 라이브러리 모듈 구성 변경 사항

Python 3.x	Python 2.x		Python 3.x
urllib.parse	urlparse	urlib 일부	하나의 urllib 패키지로 모아서 모듈을 기능별로 나눔
urllib.request	urllib 대부분		
urllib.error	urllib2 에러 부분		
urllib.response			
urllib.robotparser	Robotparser		
http.server	BaseHTTPServer		하나의 http 패키지로 모아 server와 client 모듈로 나눔
http.server	CGIHTTPServer		
http.server	SimpleHTTPServer		
http.client	httplib		
http.cookies	Cookie		하나의 http 패키지로 병합
http.cookiejar	cookielib		
html.parse	HTMLParser		하나의 http 패키지로 병합
html.entities	htmlentitydefs		

Web Server Program 기초

- HTTPServer class :
  - server address, port정보를 이용하여 server program 실행
    - BaseHTTPRequestHandler의 자식 class인 HelloHandler class 실행
- BaseHTTPRequestHandler class : web client 요청 처리
  - do_GET 메소드는 web client의 GET method 요청을 처리
    - Response header 정보 setting
    - Response body에 해당하는 html문서등을 생성 (self.wfile.write 이용)

```
from http.server import HTTPServer, BaseHTTPRequestHandler

class HelloHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/plain; charset=utf-8')
        self.end_headers()
        self.wfile.write("Hello, HTTP!₩n".encode())

if __name__ == '__main__':
    server_address = ('', 8000)  # Serve on all addresses, port 8000.
    httpd = HTTPServer(server_address, HelloHandler)
    httpd.serve_forever()
```

Web Server Program 기초

- Web server 프로그램에서 사용 가능한 주요 class
  - HTTPServer에서 이용 가능한 class는 BaseHTTPRequestHandler, SimpleHTTPRequestHandler, CGIHTTPRequestHandler등이 있음

클래스명	주요 기능
HTTPServer	• 웹 서버를 만들기 위한 클래스로, 서버 IP와 PORT를 바인딩함 • HTTPServer 객체 생성 시, 핸들러 클래스가 반드시 필요함
BaseHTTPRequestHandler	• 핸들러를 만들기 위한 기반 클래스로, HTTP 프로토콜 처리 로직이 들어있음 • 이 클래스를 상속받아, 자신의 로직 처리를 담당하는 핸들러 클래스를 만듦
SimpleHTTPRequestHandler	• BaseHTTPRequestHandler 클래스를 상속받아 만든 클래스 • GET과 HEAD 메소드 처리가 가능한 핸들러 클래스
CGIHTTPRequestHandler	• SimpleHTTPRequestHandler 클래스를 상속받아 만든 클래스 • 추가적으로 POST 메소드와 CGI 처리가 가능한 핸들러 클래스

Web Server Program 기초

- BaseHTTPRequestHandler class
  - Get 방식에서 사용하는 주요 인스턴스 변수
    - send_headers : response 객체의 header 정보에 write할 때 사용
    - end_headers : response 객체 header에 empty line 추가
    - path : request path 지정되어 있음, string data type
      - 예 : 'https://www.google.com/search?q=gray+squirrel&tbm=isch'
    - wfile : response body에 data 쓸 때 사용하는 out stream handler
  - Post 방식에서 사용하는 주요 인스턴스 변수
    - headers : request header 정보를 갖고 있는 인스턴슨 변수
    - rfile : request에서 data읽을 때 사용하는 input stream handler

urllib.parse 모듈

- urllib.parse 모듈 주요 용어
  - scheme : URL에 사용된 프로토콜을 의미
  - netloc : network 위치, HTTP protocol인 경우는 **host:port** 형식
  - path : 파일이나 애플리케이션 경로
  - params : 애플리케이션에 전달될 매개변수(현재 사용 않음)
  - query : 질의 문자열 또는 매개변수로, 앰퍼샌드(&)로 구분된 이름=값 쌍 형식으로 표현
  - fragments : 문서 내의 앵커 등 조각을 지정

```
C:₩RedBook₩ch2>python
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 18:41:36) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> from urllib.parse import urlparse
>>> result = urlparse("http://www.python.org:80/guido/python.html;philosophy?overall=3#n10")
>>> result
ParseResult(scheme='http', netloc='www.python.org:80',
path='/guido/python.html', params='philosophy', query='overall=3', fragment='n10')
```