01장 웹프로그램의 이해

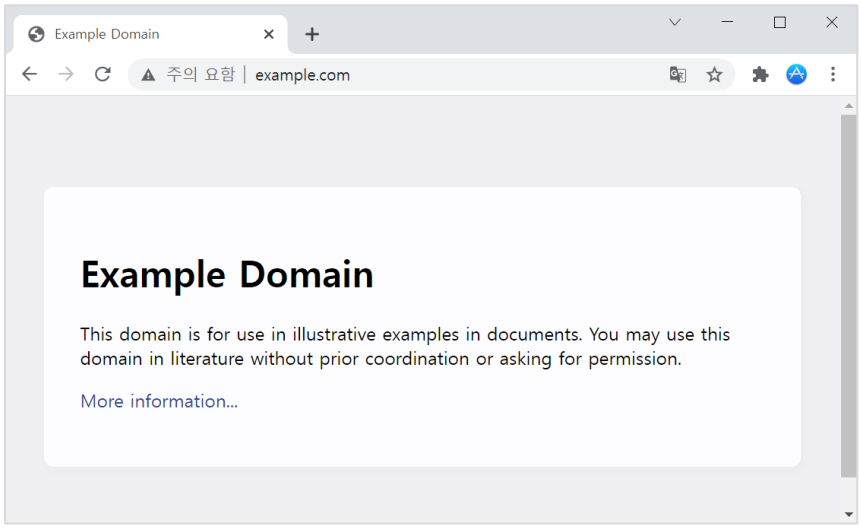
1.1 웹 프로그래밍이란?

- 간단히 말하면 HTTP(S) 프로토콜로 통신하는 클라이언트와 서버를 개발하는 것이다.

1.2 다양한 웹 클라이언트

1.2.1 웹 브라우저를 사용하는 요청

- 웹 브라우저는 주소창에 입력한 문장을 해석하여 웹 서버에게 HTTP 요청을 보내는 웹 클라이언트의 역할을 수행한다.
- 요청을 받은 웹 서버는 그 결과를 웹 브라우저로 전송해준다.



1.2.2 curl 명령을 사용하는 요청

- curl 명령은 HTTP/HTTPS/FTP 등 여러 가지의 프로토콜을 사용하여 데이터를 송수신할 수 있다.

```
D:\dev\workspace\django1\ch01>curl http://www.example.com
<!doctype html>
<html>
<head>
    <title>Example Domain</title>
    ...(생략)...
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in documents. You may use this
    domain in literature without prior coordination or asking for permission.</p>
    <p><a href="https://www.iana.org/domains/example">More information...</a></p>
```

```
</div>
</body>
</html>
```

1.2.3 Telenet을 사용하여 요청

- telnet 프로그램을 사용하여 HTTP 요청을 보낼 수도 있다.

```
D:\dev\workspace\django1\ch01>telnet www.example.com 80
```

1.2.4 직접 만든 클라이언트로 요청

- 이번에는 파이썬 프로그램으로 간단한 웹 클라이언트를 만들어 본다.

[ch01/example.py]

```
01  import urllib.request
02  print(urllib.request.urlopen("http://www.example.com").read().decode('utf-8'))
```

```
// 실행 결과
D:\dev\workspace\django1\ch01>python example.py
<!doctype html>
<html>
<head>
    <title>Example Domain</title>
    ...(생략)...
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in documents. You may use this
    domain in literature without prior coordination or asking for permission.</p>
    <p><a href="https://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
```

1.3 HTTP 프로토콜

- HTTP(Hypertext Transfer Protocol)는 웹 서버와 웹 클라이언트 사이에서 데이터를 주고받기 위해 사용하는 통신 방식으로, TCP/IP 프로토콜 위에서 동작한다.

1.3.1 HTTP 메시지의 구조

- HTTP 메시지는 클라이언트에서 서버로 보내는 요청 메시지와 서버에서 클라이언트로 보내는 응답 메시지 2가지가 있다.

1.3.2 HTTP 처리 방식

- HTTP 메소드 종류

메소드명	의미	CRUD와 매핑되는 역할
GET	리소스 취득	Read
POST	리소스 생성, 리소스 데이터 추가	Create
PUT	리소스 변경	Update
DELETE	리소스 삭제	Delete
HEAD	리소스의 헤더(메타데이터) 취득	
OPTIONS	리소스가 서포트하는 메소드 취득	
TRACE	루프백 시험에 사용	
CONNECT	프록시 동작의 터널 접속으로 변경	

1.3.3 GET와 POST 메소드

- 앞에서 8가지의 HTTP 메소드를 소개하였지만, 현실적으로 가장 많이 사용하는 메소드는 GET과 POST 2가지 이다.

1.3.4 상태 코드

- 상태 코드 분류

메소드명	의미	CRUD와 매핑되는 역할
1xx	Informational(정보제공)	임시적인 응답으로, 현재 클라이언트의 요청까지 처리되었으니 계속 진행하라는 의미이다.
2xx	Sucess	클라이언트의 요청이 서버에서 성공적으로 처리되었다는 의미이다.
3xx	Redirection	완전한 처리를 위해서 추가적인 동작을 필요로 하는 경우이다.
4xx	Client Error	없는 페이지를 요청하는 것처럼 클라이언트의 요청 메시지 내용이 잘못된 경우이다.
5xx	Server Error	서버 측 사정에 의해서 메시지 처리에 문제가 발생한 경우이다.

1.4 URL 설계

1.4.1 URL을 바라보는 측면

- URL은 웹 클라이언트에서 호출한다는 시점에서 보면 웹 서버에 존재하는 애플리케이션에 대한 API라고 할 수 있다.

1.4.2 간편 URL

- 최근에는 REST 방식의 URL 개념을 기반으로, 간단하면서도 사용자에게 친숙하게 URL을 표현하려고 노력한다.

1.4.3 파이썬의 우아한 URL

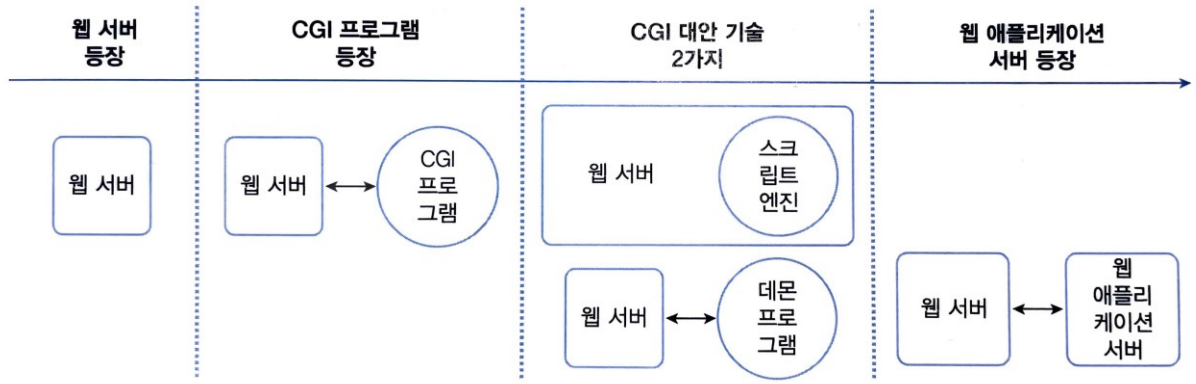
- 정규표현식을 사용하여 URL을 좀 더 구체적으로 표현하는 방식이다.

1.5 웹 애플리케이션 서버

■ 웹 서버와 웹 애플리케이션 서버 구분

구분	역할	프로그램명
웹 서버	웹 클라이언트의 요청을 받아서 요청을 처리하고, 그 결과를 웹 클라이언트에게 응답한다. 주로 정적 페이지인 HTML, 이미지, CSS, 자바스크립트 파일을 웹 클라이언트에 제공할 때 사용한다. 만약 동적 페이지 처리가 필요하다면 웹 애플리케이션 서버에 처리를 넘긴다.	Apache httpd, Nginx, lighttpd, IIS 등
웹 애플리케이션 서버	웹 서버로부터 동적 페이지 요청을 받아서 요청을 처리하고, 그 결과를 웹 서버로 반환한다. 주로 동적 페이지 생성을 위한 프로그램 실행과 데이터베이스 연동 기능을 처리한다.	Apache Tomcat, JBoss, WebLogic, WebSphere, Jetty, Jeus, mod_wsgi, uWSGI, Gunicorn 등

■ 기술의 발전에 따른 웹 서버 기술의 변화



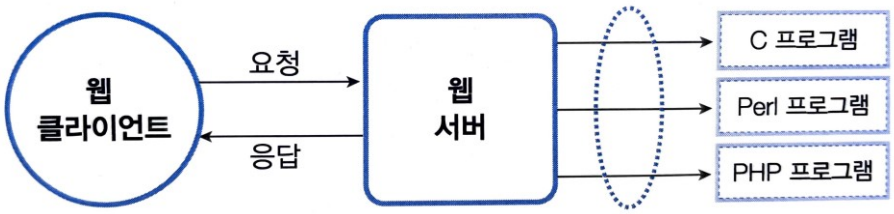
1.5.1 정적 페이지 vs 동적 페이지

■ 정적 페이지란 누가, 언제 요구하더라도 항상 같은 내용을 표시하는 웹 페이지를 말한다. 주로 HTML, 자바스크립트, CSS, 이미지만으로 이루어진 페이지가 해당된다.

■ 동적 페이지란 동일한 리소스의 요청이라도 누가, 언제, 어떻게 요구했는지에 따라 각각 다른 내용이 반환되는 페이지를 말한다. 예를 들면 현재 시각을 보여주는 페이지나 온라인 쇼핑 사이트에서 사용자마다 다른 카트 내용을 보여주는 페이지 등이 있다.

1.5.2 CGI 방식의 단점

■ 전통적인 CGI 방식의 요청 처리



독립적인 프로세스로 실행

- CGI 방식의 근본적인 문제점은 각각의 클라이언트 요청에 대하여 독립적인 별도의 프로세스가 생성된다는 것이다. 요청이 많아질수록 프로세스가 많아지고, 프로세스가 많아질수록 비례적으로 프로세스가 점유하는 메모리 요구량도 커져서 시스템에 많은 부하를 주는 요인이 된다.

1.5.3 CGI 방식의 대안 기술

- CGI 방식의 대한 기술 중 하나는 별도의 애플리케이션을 Perl, PHP 등의 스크립트 언어로 작성하고 스크립트를 처리하는 스크립트 엔진을 웹 서버에 내장시켜서 CGI 방식의 단점이었던 별도의 프로세스를 기동시키는 오버헤드를 줄이는 방식이다. 파이썬의 경우에는 mod_wsgi 모듈을 사용하고 있다.

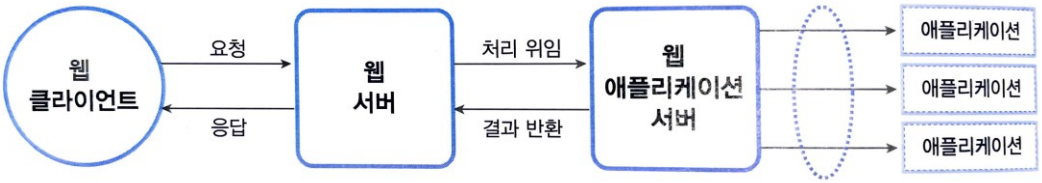
- 또 다른 방식은 애플리케이션을 처리하는 프로세스를 미리 데몬(서비스)으로 기동시켜 놓은 후, 웹 서버의 요청을 데몬에서 치리하는 것이다. 이 또한 프로세스 생성 부하를 줄일 수 있는 방법이다. 파이썬의 경우에는 데몬 방식에도 mod_wsgi 모듈을 사용한다. mod_wsgi 모듈은 앞에서 처럼 웹 서버 내장 방식으로도 실행이 가능하고, 별도의 데몬 방식으로도 실행이 가능하다.

- CGI 애플리케이션을 별도의 데몬으로 처리하는 방식은 기술이 점차 발전함에 따라, 스레드 처리가 보강되고 객체 지향 기술이 반영되면서 애플리케이션 전용 데몬인 애플리케이션 서버 방식으로 발전하였다. 현재 가장 많이 사용되고 있는 JSP, ASP 기술에서 애플리케이션 서버 방식을 사용하고 있다. 파이썬에서 웹 서버와 연동용으로 사용하는 mod_wsgi, uwsgi, gunicorn 프로그램들이 웹 서버 프로그램인 httpd, nginx와는 별개의 애플리케이션 전용 데몬으로 동작한다는 점에서 웹 애플리케이션 서버라고 얘기할 수 있다.

1.5.4 애플리케이션 서버 방식

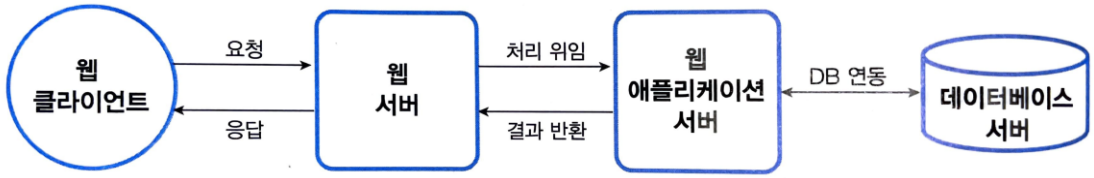
- 애플리케이션 서버 방식은 웹 서버가 직접 프로그램을 호출하기보다는 웹 애플리케이션 서버를 통해서 간접적으로 웹 애플리케이션 프로그램을 실행한다.

- 애플리케이션 서버 방식의 요청 처리



웹 애플리케이션
서버에서 실행

■ 애플리케이션 서버 방식에서의 서버 간 구성도



1.5.5 웹 서버와의 역할 구분

■ 웹 서버와 웹 애플리케이션 서버의 역할과 HW 배치

- 거의 모든 웹 사이트가 정적 페이지와 동적 페이지를 같이 제공하는 환경에서 웹 서버 또는 웹 애플리케이션 서버 하나만으로 서비스하는 것은 매우 비효율적이다.
- 하나의 HW 박스에 웹 서버와 웹 애플리케이션 서버를 모두 탑재하는 것도 가능하지만, HW 박스를 분리하여 구성하면 메모리 효율을 더욱 더 높일 수 있다.