

▶ CHAPTER 01 웹 프로그래밍의 이해

파이썬 웹프로그래밍

시작하기 전에

- 개발 환경

- Django 2.0
- Python 3.6
- Windows 10
- 장고 공식 홈페이지
<https://docs.djangoproject.com/>
(오픈소스)

- 예제 파일 다운로드

- <http://www.hanbit.co.kr/src/10104>

이 책의 학습 목표

- CHAPTER 01 프로그래밍의 기본 기술에 대해 이해하기
- CHAPTER 02 파이썬 웹 라이브러리의 전체 구성 및 버전 2.x와 3.x의 달라진 점
- CHAPTER 03 Django 장고를 사용하여 웹 애플리케이션을 만드는 과정
- CHAPTER 04 장고의 기능 중에서 실제 프로젝트 개발을 위해 꼭 알아야 되는 6개 기능
- CHAPTER 05 클래스형 뷰를 사용하는 애플리케이션 만들기
- CHAPTER 06 웹 서버로 오래전부터 사용된 아파치와 차세대 웹 서버로 알려진 NGINX에 대해 알아보기
- CHAPTER 07 파이썬 개발자들이 많이 사용하는 PythonAnywhere 사이트에 대해 알아보기
- CHAPTER 08 아파치 웹 서버에서 장고 프로그램을 연동하기 위해 사용하는 mod_wsgi 프로그램 설정 및 실행
- CHAPTER 09 NGINX의 설치 방법과 장고 프로그램을 실행하기 위한 설정 방법

Contents

- CHAPTER 01 웹 프로그래밍의 이해
 - 1.1 웹 프로그래밍이란?
 - 1.2 다양한 웹 클라이언트
 - 1.2.1 웹 브라우저를 사용하여 요청
 - 1.2.2 리눅스 curl 명령을 사용하여 요청
 - 1.2.3 Telnet을 사용하여 요청
 - 1.2.4 직접 만든 클라이언트로 요청
 - 1.3 HTTP 프로토콜
 - 1.3.1 HTTP 메시지의 구조
 - 1.3.2 HTTP 처리 방식
 - 1.3.3 GET과 POST 메소드
 - 1.3.4 상태 코드

Contents

- CHAPTER 01 웹 프로그래밍의 이해
 - 1.4 URL 설계
 - 1.4.1 URL을 바라보는 측면
 - 1.4.2 간편 URL
 - 1.4.3 파이썬의 우아한 URL
 - 1.5 웹 애플리케이션 서버
 - 1.5.1 정적 페이지 vs 동적 페이지
 - 1.5.2 CGI 방식의 단점
 - 1.5.3 CGI 방식의 대안 기술
 - 1.5.4 애플리케이션 서버 방식
 - 1.5.5 웹 서버와의 역할 구분



CHAPTER 01 웹 프로그래밍의 이해

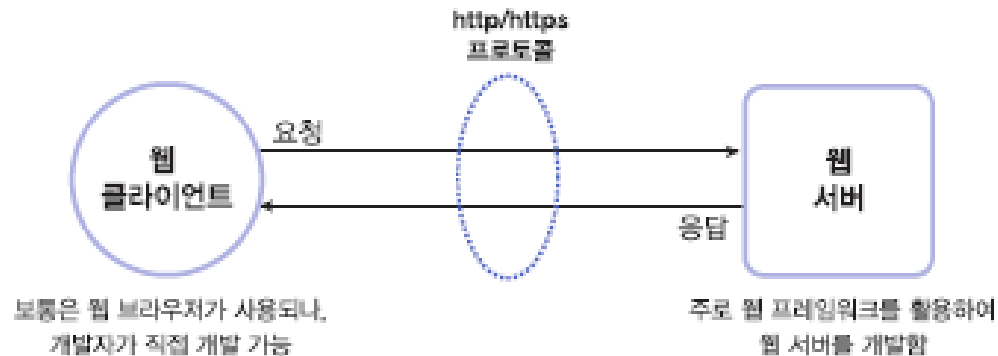
웹 프로그래밍의 기본 기술에 대해 이해하기



HTTP(S) 프로토콜로 통신하는, 클라이언트와 서버를 개발

웹 클라이언트와 웹 서버를 같이 개발 및 웹 클라이언트 또는 웹 서버 하나만 개발

웹 서버를 개발하는 경우가 많아서 파이썬 웹 프로그래밍이라고 하면 우선적으로 장고^{Django}와 같은 웹 프레임워크를 사용하여 웹 서버를 개발





1.2.1 웹 브라우저를 사용하여 요청



브라우저는 주소창에 입력된 문장을 해석하여 웹 서버에게 HTTP 요청을 보내는 웹 클라이언트의 역할을 수행함

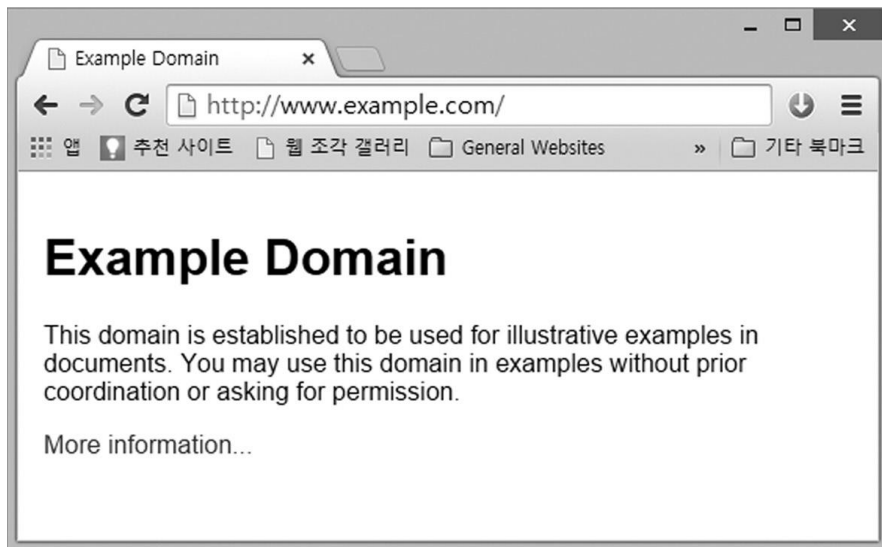


그림 1-3 웹 브라우저 요청에 대한 웹 서버의 응답



1.2.2 리눅스 curl 명령을 사용하여 요청

리눅스 **curl** 명령은 HTTP/HTTPS/FTP 등 여러 가지의 프로토콜을 사용하여 데이터를 송수신할 수 있는 명령

```
$ curl http://www.example.com|
```

```
[shkim@localhost ch1]$ curl http://www.example.com
<!doctype html>
<html>
<head>
  <title>Example Domain</title>
  . . . (중략)

</head>

<body>
<div>
  <h1>Example Domain</h1>
  <p>This domain is established to be used for illustrative examples in
documents. You may use this domain in examples without prior coordination or
asking for permission.</p>
  <p><a href="http://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
[shkim@localhost ch1]$
```

그림 1-4 curl 명령 요청에 대한 웹 서버의 응답

[그림 1-3]과 비교해서 HTML 태그들과 <head> 부분이 더 보이지만 이는 동일한 응답



1.2.3 Telnet을 사용하여 요청

리눅스의 **telnet** 프로그램을 사용하여 HTTP 요청을 보낼 수도 있음

```
[shkim@localhost ch1]$ telnet www.example.com 80
Trying 93.184.216.119...
Connected to www.example.com.
Escape character is '^]'.
GET / HTTP/1.1  입력
Host: www.example.com  입력
 입력
```

telnet 명령은 터미널 창에서 입력하는 내용을 그대로 웹 서버에 전송

telnet 프로그램이 웹 클라이언트의 역할을 수행

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html
Date: Sun, 09 Nov 2014 11:46:41 GMT
Etag: "359670651"
Expires: Sun, 16 Nov 2014 11:46:41 GMT
Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Server: ECS (rhv/B18F)
X-Cache: HIT
X-ec-custom-error: 1
Content-Length: 1270

<!doctype html>
<html>
<head>
  <title>Example Domain</title>
  ... (중략)

</head>
<body>
<div>

  <h1>Example Domain</h1>
  <p>This domain is established to be used for illustrative examples in
  documents. You may use this domain in examples without prior coordination or
  asking for permission.</p>
  <p><a href="http://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
^]
telnet> quit  입력
Connection closed.
[shkim@localhost ch1]$
```

그림 1-5 telnet 명령 요청에 대한 웹 서버의 응답

응답 메시지를 수신한 후, **telnet** 프로그램을 종료하기 위해 마지막 두 라인을 입력한 것에유의.
동일한 응답을 확인할 수 있음



1.2.4 직접 만든 클라이언트로 요청

```
C:\WRedBook\wch1>notepad example.py

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

C:\WRedBook\wch1>python example.py
<!doctype html>
<html>
<head>
  <title>Example Domain</title>
  ... (중략)
</head>

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

C:\WRedBook\wch1>
```

그림 1-6 직접 만든 클라이언트 요청에 대한 웹 서버의 응답

앞에서 보았던 응답 결과와 동일한 것을 확인할 수 있음.
즉, 웹 클라이언트의 형태는 달라도 동일한 요청에 대해서 동일한 응답을 받는 것을 확인

```
C:\WRedBook\wch1>python -c "import urllib.request; print(urllib.request.urlopen('http://
www.example.com').read().decode('utf-8'))"
```

웹 클라이언트를 직접 만드는 방법은 **2.2** 웹 클라이언트 라이브러리에서 자세히 설명하고 있으니, 참 고 바랍니다



1.3.1 HTTP 메시지의 구조

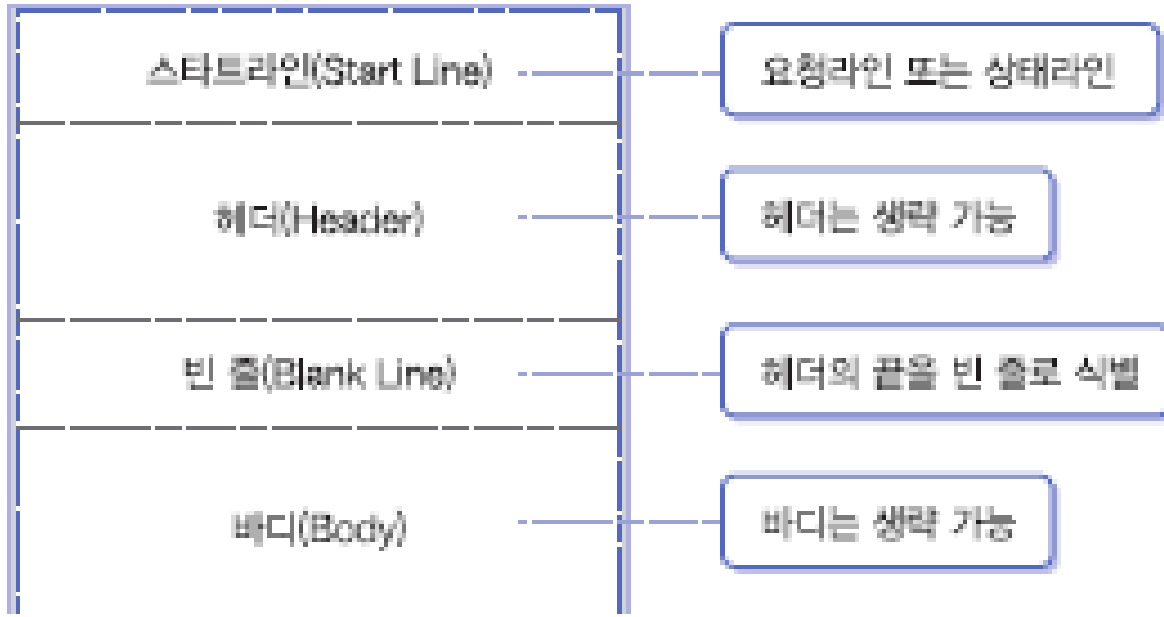


그림 1-7 HTTP 메시지 구조

[그림 1-7]에서 스타트라인은 요청 메시지일 때 요청라인_{request line}이라고 하고, 응답 메시지일 때 상태라인_{status line}이라고 함

```
GET /book/shakespeare HTTP/1.1
Host: www.example.com:8080
```

첫 번째 줄은 요청라인으로, 요청 방식^{method},
요청 **URL**, 프로토콜 버전으로 구성

NOte

URI vs URL란? URI는 Uniform Resource Identifier의 약자로
URL(Uniform Resource Locator)과 URN(Uniform Resource Name)을 포함하
는 좀 더 넓은 의미의 표현이지만, 웹 프로그래밍에서는
URI와 URL을 동일한 의미로 사용해도 무방



1.3.2 HTTP 처리 방식

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

표 1-1 HTTP 메소드 종류

GET 방식은 지정한 URL의 정보를 가져오는 메소드로, 가장 많이 사용

POST의 대표적인 기능은 리소스를 생성하는 것으로, 블로그에 글을 등록하는 경우가 이에 해당.

PUT은 리소스를 변경하는 데 사용



1.3.3 GET과 POST 메소드

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

```
GET http://docs.djangoproject.com/search/?q=forms&release=1 HTTP/1.1
```

```
POST http://docs.djangoproject.com/search/ HTTP/1.1  
Content-Type: application/x-www-form-urlencoded
```

```
q=forms&release=1
```

파이썬의 장고 프레임워크에서도 폼의 데이터는
POST 방식만을 사용하고 있음



그림 1-8 네이버의 검색 창 - GET 방식 전달



1.3.4 상태 코드

서버에서의 처리 결과는 응답 메시지의 상태라인에 있는
상태 코드Status code를 보고 파악할 수 있음

메소드명	의미	CRUD와 매핑되는 역할
1xx	Informational (정보 제공)	임시적인 응답으로, 현재 클라이언트의 요청까지 처리되었으니 계속 진행하라는 의미입니다. HTTP 1.1 버전부터 추가되었습니다.
2xx	Success(성공)	클라이언트의 요청이 서버에서 성공적으로 처리되었다는 의미입니다.
3xx	Redirection (리다이렉션)	완전한 처리를 위해서 추가적인 동작을 필요로 하는 경우입니다. 주로 서버의 주 소 또는 요청한 URI의 웹 문서가 이동되었으니, 그 주소로 다시 시도해보라는 의 미입니다.
4xx	Client Error (클라이언트 에러)	없는 페이지를 요청하는 것처럼 클라이언트의 요청 메시지 내용이 잘못된 경우 입니다.
5xx	Server Error (서버 에러)	서버 측 사정에 의해서 메시지 처리에 문제가 발생한 경우입니다. 서버의 부하 DB 처리 과정 오류, 서버에서 익셉션이 발생하는 경우가 이에 해당합니다.

표 1-2 상태 코드 분류



1.3.4 상태 코드

상태 코드	상태 텍스트	응답 문구	서버 측면에서의 의미
2xx	Success	성공	클라이언트가 요청한 동작을 수신하여 이해했고, 승낙했으며 성공적으로 처리했다.
200	OK	성공	서버가 요청을 성공적으로 처리했다.
201	Created	생성됨	요청이 처리되어서 새로운 리소스가 생성되었다. 응답 헤더 Location에 새로운 리소스의 절대 URI를 기록합니다.
202	Accepted	허용됨	요청은 접수했지만 처리가 완료되지 않았다. 클라이언트는 응답 헤더의 Location, Retry-After를 참고하여 다시 요청을 보냅니다.
3xx	Redirection	리다이렉션	클라이언트는 요청을 마치기 위해 추가적인 동작을 취해야 한다.
301	Moved Permanently	영구 이동	지정된 리소스가 새로운 URI로 이동했다. 이동할 곳의 새로운 URI는 응답 헤더 Location에 기록합니다.

표 1-3 자주 사용되는 상태 코드



1.3.4 상태 코드

상태 코드	상태 텍스트	응답 문구	서버 측면에서의 의미
303	See Other	다른 위치 보기	다른 위치로 요청하라. 요청에 대한 처리 결과를 응답 헤더 Location에 표시된 URI에서 GET으로 취득할 수 있습니다. 브라우저의 폼 요청을 POST로 처리하고 그 결과 화면으로 리다이렉트시킬 때 자주 사용하는 응답 코드입니다.
307	Temporary Redirect	임시 리다이렉션	임시로 리다이렉션 요청이 필요하다. 요청한 URI가 없으므로, 클라이언트는 메소드를 그대로 유지한 채 응답 헤더 Location에 표시된 다른 URI로 요청을 재송신할 필요가 있습니다. 클라이언트는 향후 요청 시 원래 위치를 계속 사용해야 합니다. 302의 의미를 정확하게 재정의해서 HTTP/1.1의 307 응답으로 추가되었습니다.
4xx	Client Error	클라이언트 에러	클라이언트의 요청에 오류가 있다.
400	Bad Request	잘못된 요청	요청의 구문이 잘못되었다. 클라이언트가 모르는 4xx 계열의 응답 코드가 반환된 경우에도 클라이언트는 400과 동일하게 처리하도록 규정하고 있습니다.
401	Unauthorized	권한 없음	지정한 리소스에 대한 액세스 권한이 없다. 응답 헤더 WWW-Authenticate에 필요한 인증 방식을 지정합니다.
403	Forbidden	금지됨	지정한 리소스에 대한 액세스가 금지되었다. 401 인증 처리 이외의 사유로 리소스에 대한 액세스가 금지되었음을 의미합니다. 리소스의 존재 자체를 문제하고 싶은 경우는 404 응답 코드를 사용할 수 있습니다.
404	Not Found	찾을 수 없음	지정한 리소스를 찾을 수 없다.
5xx	Server Error	서버 에러	클라이언트의 요청은 유효한데, 서버가 처리에 실패했다.
500	Internal Server Error	내부 서버 오류	서버쪽에서 에러가 발생했다. 클라이언트가 모르는 5xx 계열의 응답 코드가 반환된 경우에도 클라이언트는 500과 동일하게 처리하도록 규정하고 있습니다.
502	Bad Gateway	불량 게이트웨이	게이트웨이 또는 프록시 역할을 하는 서버가 그 뒷단의 서버로부터 잘못된 응답을 받았다.
503	Service Unavailable	서비스 제공불가	현재 서버에서 서비스를 제공할 수 없다. 보통은 서버의 과부하나 서비스 점검 등 일시적인 상태입니다.



그림 1-9 URL 구성 항목

URL 스킴 : URL에 사용된 프로토콜을 의미.

- 호스트명 : 웹 서버의 호스트명으로, 도메인명 또는 IP 주소로 표현.
- 포트번호 : 웹 서버 내의 서비스 포트번호. 생략 시에는 디폴트 포트번호로, http는 80을, https는 443을 사용
- 경로 : 파일이나 애플리케이션 경로를 의미
- 쿼리스트링 : 질의 문자열로, 앰퍼샌드(&)로 구분된 이름=값 쌍 형식으로 표현
- 프래그먼트 : 문서 내의 앵커 등 조각을 지정



1.4.1 URL을 바라보는 측면

URL은 웹 클라이언트에서 호출한다는 시점에서 보면, 웹 서버에 존재하는 애플리케이션에 대한 **API**_{Application Programming Interface}라고 할 수 있음

```
http://blog.example.com/search?q=test&debug=true
```

URL을 바라보는 또 한 가지 측면은 REST 방식으로 URL을 설계하는 것.
REST 방식이란 웹 서버에 존재하는 요소들을 모두 리소스라고 정의하고, URL을 통해 웹 서버의 특정 리소스를 표현한다는 개념.



1.4.2 간편 URL

간편 URL은 쿼리스트링 없이 경로만 가진 간단한 구조의 URL을 말함.

검색 엔진의 처리를 최적화하기 위해 생겨난 간편한 URL은 URL을 입력하거나 기억하기 쉽다는 부수적인 장점도 있어, 검색 엔진 친화적 URL_{search engine friendly url} 또는 사용자 친화적 URL_{user friendly url}이라고 부르기도 함

기존 URL	간편 URL
<code>http://example.com/index.php?page=foo</code>	<code>http://example.com/foo</code>
<code>http://example.com/index.php?page=consulting/marketing</code>	<code>http://example.com/consulting/marketing</code>
<code>http://example.com/products?category=2&pid=25</code>	<code>http://example.com/products/2/25</code>
<code>http://example.com/cgi-bin/feed.cgi?feed=news&frm=rss</code>	<code>http://example.com/news.rss</code>
<code>http://example.com/services/index.jsp?category=legal&id=patents</code>	<code>http://example.com/services/legal/patents</code>
<code>http://example.com/index.asp?mod=profiles&id=193</code>	<code>http://example.com/user/john-doe</code>
<code>http://example.com/app/dashboard/dsptchr_c80.dll?page=38661&mod1=bnr_ant&UID=4511681&SESSIONID=4fd8b561ac867195fba2cc5679&...</code>	<code>http://example.com/app/dashboard/reports#monthly</code>

표 1-4 기존 URL과 간편 URL 간의 비교를 위한 대응 예시



1.4.3 파이썬의 우아한 URL

URL을 정의하기 위해 정규표현식 Regular Expression 을 추가적으로 사용할 수 있음.

```
urlpatterns = [
    path('articles/2003/', views.special_case_2003),
    path('articles/<int:year>/', views.year_archive),
    path('articles/<int:year>/<int:month>/', views.month_archive),
    path('articles/<int:year>/<int:month>/<slug:slug>/', views.article_detail),
]

urlpatterns = [
    path('articles/2003/', views.special_case_2003),
    re_path(r'^articles/(?P<year>[0-9]{4})/$', views.year_archive),
    re_path(r'^articles/(?P<year>[0-9]{4})/(?P<month>[0-9]{2})/$', views.month_archive),
    re_path(r'^articles/(?P<year>[0-9]{4})/(?P<month>[0-9]{2})/(?P<slug>[ww-]+)/$', views.
article_detail),
]
```

SECTION 01_5 웹 애플리케이션 서버



구분	역할	프로그램 명
웹 서버	<p>웹 클라이언트의 요청을 받아서 요청을 처리하고, 그 결과를 웹 클라이언트에게 응답합니다.</p> <p>주로 정적 페이지인 HTML, 이미지, CSS, 자바스크립트 파일을 웹 클라이언트에 제공할 때 웹 서버를 사용합니다. 만약 동적 페이지 처리가 필요하다면 웹 애플리케이션 서버에 처리를 넘깁니다.</p>	Apache httpd, Nginx, lighttpd, IIS 등
웹 애플리케이션 서버	<p>웹 서버로부터 동적 페이지 요청을 받아서 요청을 처리하고, 그 결과를 웹 서버로 반환합니다.</p> <p>주로 동적 페이지 생성을 위한 프로그램 실행과 데이터베이스 연동 기능을 처리합니다.</p>	Apache Tomcat, JBoss, WebLogic, WebSphere, Jetty, Jeus, mod_wsgi, uWSGI, Gunicorn 등

웹 서버 및 웹 애플리케이션 서버라는 용어는 SW 측면의 서버 프로그램을 의미합니다.
HW 측면의 용어는 **1.5.5** 웹 서버와의 역할 구분에서 설명

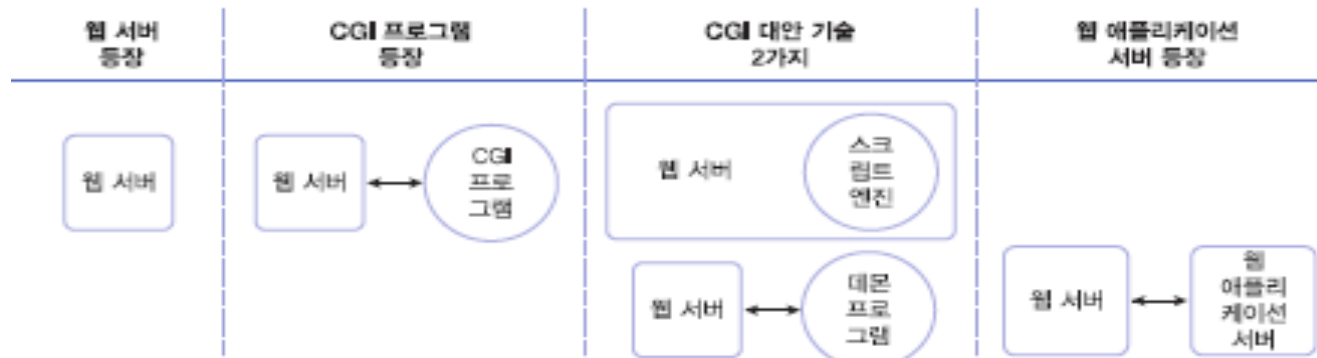


그림 1-10 기술의 발전에 따른 웹 서버 기술의 변화



1.5.1 정적 페이지 vs 동적 페이지

정적_{static}, 동적_{dynamic}이란 용어는 사용자가 페이지를 요청하는 시점에 페이지의 내용이 유지되는가 또는 변경되는가를 구분해주는 용어.

동적 페이지에는 프로그래밍 코드가 포함되어 있어서 페이지 요청 시점에 HTML 문장을 만들어내는 것



1.5.2 CGI 방식의 단점

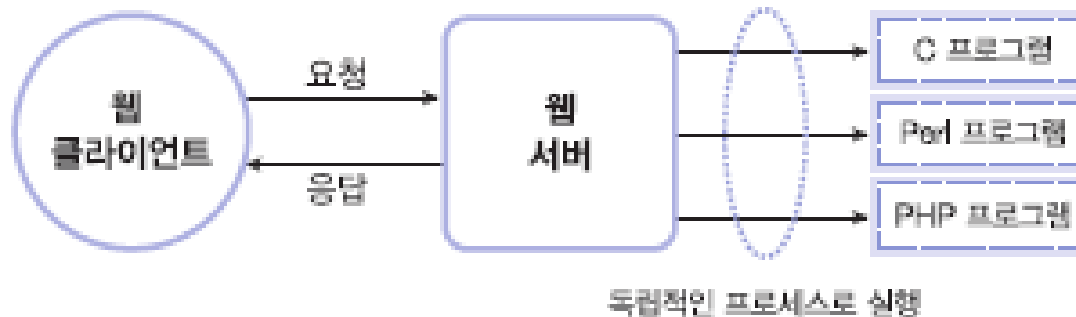


그림 1-11 전통적인 CGI 방식의 요청 처리

CGI 방식의 근본적인 문제점은 각각의 클라이언트 요청에 대하여 독립적인 별도의 프로세스가 생성

요청이 많아질수록 프로세스가 많아지고, 프로세스가 많아질수록 비례적으로 프로세스가 점유하는 메모리 요구량도 커져서 시스템에 많은 부하를 주는 요인



1.5.3 CGI 방식의 대안 기술

CGI 방식의 대안 기술 중 하나는 별도의 애플리케이션(CGI 프로그램과 같은 역할을 하는 프로그램)을 Perl, PHP 등의 스크립트 언어로 작성하고, 스크립트를 처리하는 스크립트 엔진(인터프리터)을 웹 서버에 내장시켜서 CGI 방식의 단점이었던 별도의 프로세스를 기동시키는 오버헤드를 줄이는 방식

1.5.4 애플리케이션 서버 방식

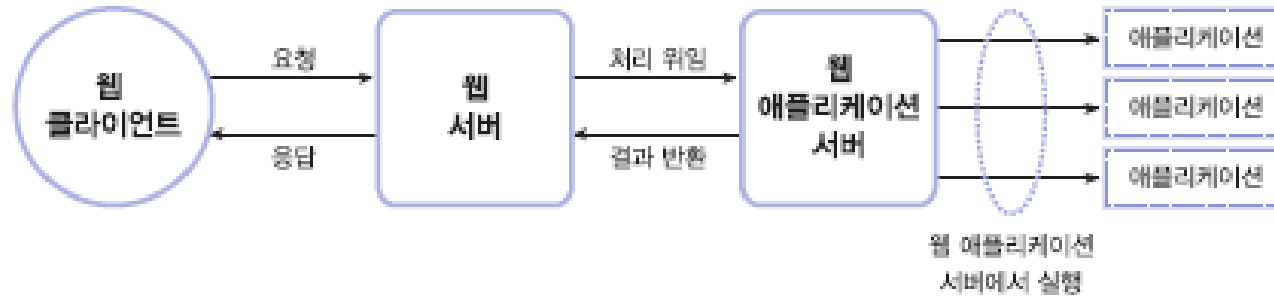


그림 1-12 애플리케이션 서버 방식의 요청 처리

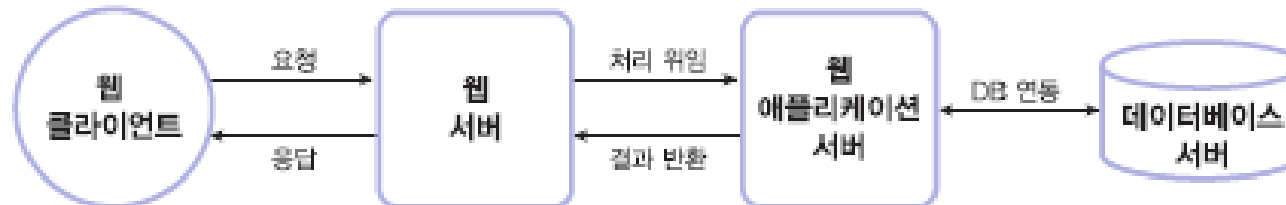


그림 1-13 애플리케이션 서버 방식에서의 서버 간 구성도

웹 서버와 웹 애플리케이션 서버가 분리됨에 따라, 서로의 역할도 구분하여 사용하는 것이 좋음. 왜냐하면 정적 페이지를 처리하는 경우에 비해서 동적 페이지를 처리하는 경우가 수 배에서 수십 배의 메모리를 소비하기 때문.



1.5.5 웹 서버와의 역할 구분

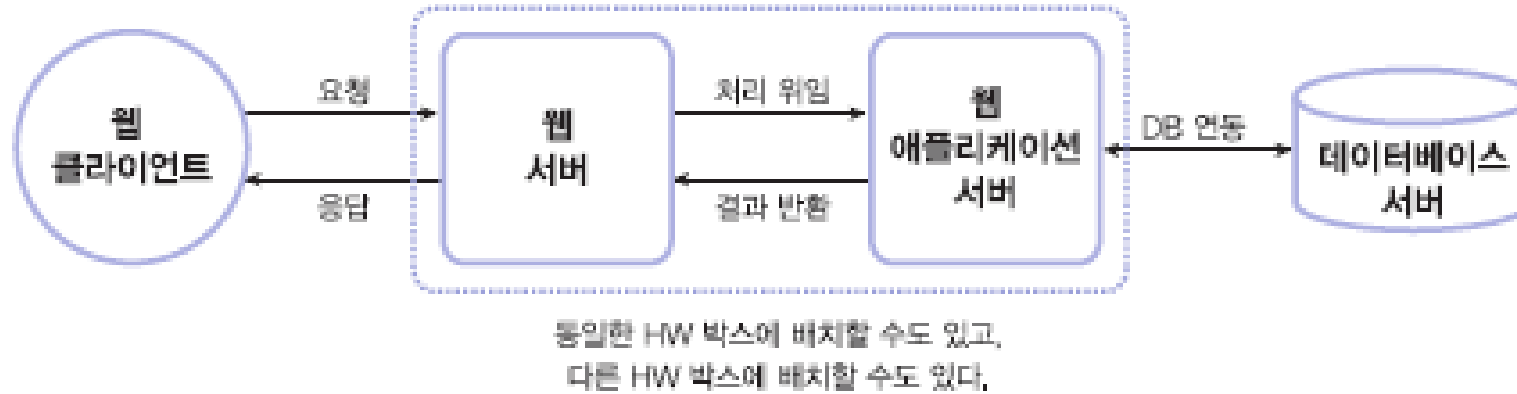


그림 1-14 웹 서버와 애플리케이션 서버의 역할과 HW 배치

지금까지 웹 서버 및 웹 애플리케이션 서버라는 용어를 사용하였는데, 이는 SW_{Software} 측면의 서버 프로그램을 의미

웹 서버와 웹 애플리케이션 서버 프로그램이 함께 필요하며, 이 두 개의 서버를 동일한 HW 박스에서 기동시키는 것도 충분히 가능한 구성.
서비스 운용 관리 측면에서 하나의 HW 박스에 구성하는 것이 좀 더 간편한 방식이기 때문