



HTTP와 웹 아키텍처 (ver1.0.1)

# 목 차

---

1. 웹 개요
2. HTTP 둘러보기
3. HTTP 이해
4. 웹 서버와 웹 아키텍처
5. 웹 애플리케이션 아키텍처
6. 웹 애플리케이션 트렌드





# 1. 웹(Web) 소개

---

- 1.1 웹이란?
- 1.2 웹의 등장
- 1.3 사용자 관점에서 보는 웹
- 1.4 요약

## 1.1 웹이란?

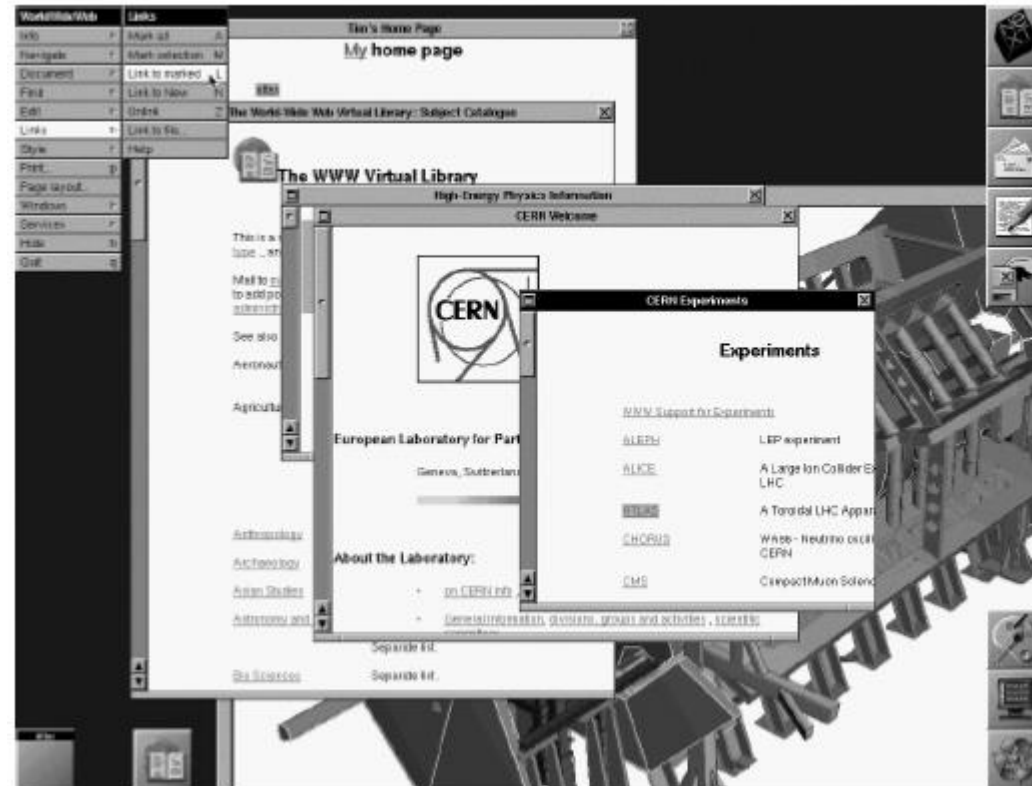
- ✓ 웹은 월드와이드웹(World Wide Web)의 줄임 말로, 전 세계의 정보를 공유할 수 있는 시스템 구조입니다.
- ✓ 인터넷에 연결되어 있는 컴퓨터라면 웹을 통해 손쉽게 전 세계의 지식과 자원을 주고 받을 수 있습니다.
- ✓ 근본적인 웹의 구조를 이해하는 것은 웹 애플리케이션을 설계하고 개발하는데 있어 탄탄한 밑거름이 될 것입니다.



출처 : <http://mobilemarketingmagazine.com/wp-content/uploads/2015/08/Connected-Globe-Internet.jpg>

## 1.2 웹의 등장

- ✓ 웹의 개념은 1989년 3월 CERN(유럽 입자 물리학 연구소)에서 처음 등장했습니다.
- ✓ 팀 버너스는 멀리 떨어져 있는 연구원들과의 지식 공유를 위해 웹을 고안하게 되었습니다.
- ✓ 월드와이드웹은 최초의 웹 브라우저 이름이며 후에 웹 시스템과 구분하기 위해 넥서스(Nexus)로 이름을 변경합니다.
- ✓ 대표적인 웹의 구성 요소로서 HTML, HTTP 그리고 URL이 세 가지가 제안되었습니다.

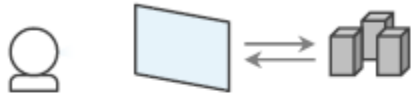


[https://www.w3.org/History/1994/WWW/Journals/CACM/screensnap2\\_24c.gif](https://www.w3.org/History/1994/WWW/Journals/CACM/screensnap2_24c.gif)

## 1.3 사용자 관점에서 보는 웹

- ✓ 사용자는 웹 브라우저를 실행하고 웹주소를 입력 합니다.
- ✓ 그 후 사용자는 다양한 메뉴와 링크를 통해 원하는 정보를 확인합니다.
- ✓ 이 같은 간단한 웹의 접근을 통해 웹을 구성하는 요소를 파악할 수 있습니다.
- ✓ 웹의 구성은 다음과 같습니다.

### 1 웹 브라우저



사용자는 웹 브라우저를 통해 웹을 경험합니다.  
사용자의 이벤트에 따라 서버에 요청을 보내고, 응답을 받아 다시 사용자에게 보여 줍니다.


### 3 하이퍼텍스트



웹 페이지는 순서가 없습니다. 사용자의 선택에 따라 변경되고, 다른 문서로 연결됩니다. 이러한 문서 작성 기법을 하이퍼텍스트(Hypertext) 라고 하고, 이 문서를 작성하기 위한 언어를 HTML이라고 부릅니다.



### 2 URL

 `http://www.nextree.co.kr/p11927/`

정보에 찾아가려면 주소를 입력해야 합니다.  
이를 URL(Uniform Resource Locator)라고 합니다.  
즉, 웹에 표시되는 모든 자원은 주소를 가지고 있습니다.

### 4 HTTP



URL의 가장 앞 부분에 HTTP (Hyper Text Transfer Protocol) 가 등장합니다. 웹은 HTTP라는 이름의 하이퍼텍스트 문서 전송 규칙들을 이용해 통신합니다.

## 1.4 요약

- ✓ 웹은 월드와이드웹(World Wide Web)의 줄임 말로서, 전 세계의 정보를 공유할 수 있는 시스템 구조입니다.
- ✓ 팀 버너스 리 박사가 제안한 웹은 HTML, URL, HTTP로 구성됩니다.
- ✓ 사용자는 웹 브라우저를 통해 웹 시스템을 사용하며, 브라우저는 HTML 문서 등을 이용해 웹 페이지를 표현합니다.
- ✓ HTTP는 요청과 응답에 대한 데이터를 전송하는 규칙을 의미하며, URL을 이용해 요청한 자원을 쉽게 탐색할 수 있습니다.

### 1 웹 브라우저




사용자는 웹 브라우저를 통해 웹을 경험합니다.  
사용자의 이벤트에 따라 서버에 요청을 보내고, 응답을 받아 다시 사용자에게 보여 줍니다.

### 3 하이퍼텍스트



웹 페이지는 순서가 없습니다. 사용자의 선택에 따라 변경되고, 다른 문서로 연결됩니다. 이러한 문서 작성 기법을 하이퍼텍스트(Hypertext)라고 하고, 이 문서를 작성하기 위한 언어를 HTML이라고 부릅니다.

### 2 URL



`http://www.nextree.co.kr/p11927/`

정보에 찾아가려면 주소를 입력해야 합니다.  
이를 URL(Uniform Resource Locator)라고 합니다.  
즉, 웹에 표시되는 모든 자원은 주소를 가지고 있습니다.

### 4 HTTP



URL의 가장 앞 부분에 HTTP (Hyper Text Transfer Protocol)가 등장합니다. 웹은 HTTP라는 이름의 하이퍼텍스트 문서 전송 규칙들을 이용해 통신합니다.







## 2. HTTP 둘러보기

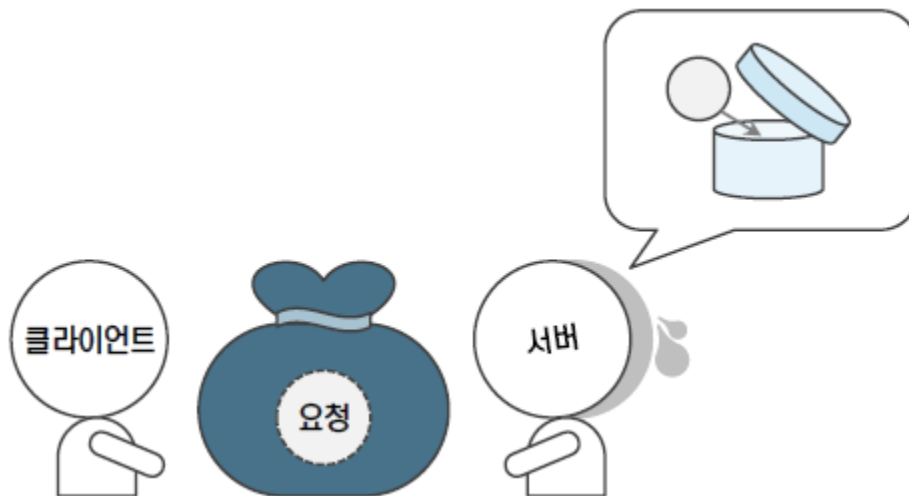
---

- 2.1 HTTP 소개
- 2.2 요청에서 응답까지
- 2.3 Stateless HTTP
- 2.4 TCP/IP와 HTTP
- 2.5 요약



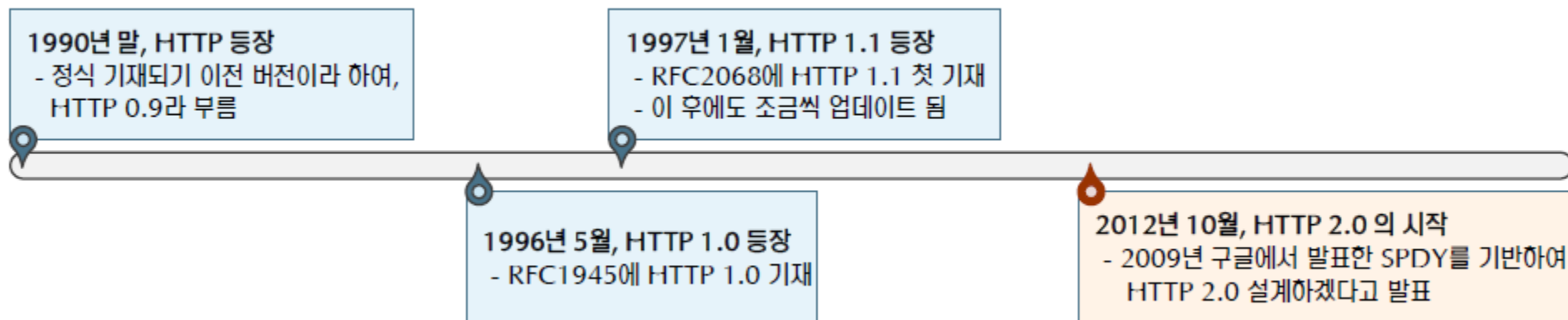
## 2.1 HTTP 소개(1/2) – 프로토콜

- ✓ HTTP(Hyper-Text Transfer Protocol)은 하이퍼텍스트 문서를 전송하기 위한 통신 규약입니다.
- ✓ 프로토콜(Protocol)이란 클라이언트와 서버간의 통신을 위해서 서로 협의해야 하는 규칙을 의미합니다.
- ✓ 데이터를 송신하는 쪽과 수신하는 쪽 모두 데이터를 주고 받는 방식에 대한 약속을 지켜야 합니다.
- ✓ 웹을 통한 통신에는 다양한 프로토콜이 존재하며, HTTP는 웹의 기반이 되는 프로토콜입니다.



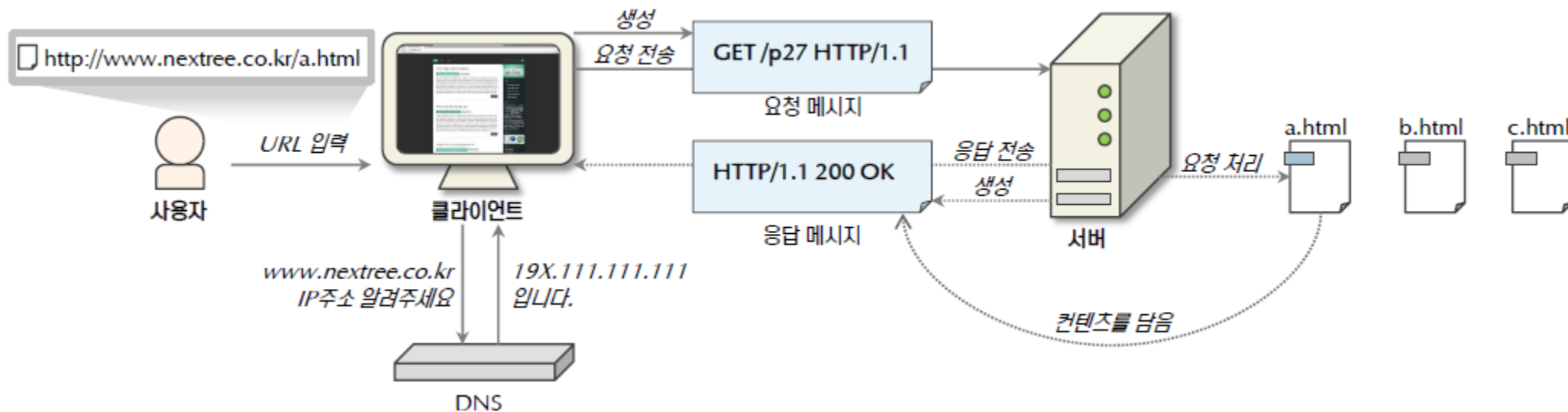
## 2.2 HTTP 소개(2/2) – HTTP의 역사

- ✓ 1995년 5월, 인터넷 표준 문서인 RFC(Request For Comments)에 HTTP/1.0이 정식으로 기재되었습니다.
- ✓ HTTP는 요청과 응답의 연속입니다. 따라서 이전 요청에 대한 응답이 도착하지 않으면 새로운 요청을 할 수 없습니다.
- ✓ 이와 같은 회전 지연(latency)으로 인해 성능이 저하되는 문제점을 해결하기 위해 HTTP/2.0이 등장 했습니다.
- ✓ HTTP/2.0은 클라이언트가 매번 요청하지 않아도 서버가 필요한 리소스들을 보내는 서버 푸시 방식을 도입했습니다.
- ✓ 하지만 현재에도 HTTP/1.1이 가장 널리 사용되고 있습니다.



## 2.2 요청에서 응답까지

- ✓ HTTP는 클라이언트와 서버간의 커뮤니케이션을 가능하게 합니다.
- ✓ 클라이언트는 사용자가 입력한 주소(URL)를 기반으로 웹 서버에 연결 합니다.
- ✓ HTTP 요청 메시지 규격에 맞게 자원의 주소와 요청 방식 들을 메시지에 담아 서버에 요청을 보냅니다.
- ✓ 서버는 요청에 대한 자원과 처리 결과 코드를 응답 메시지에 담아 반환하고, 커넥션을 닫습니다.

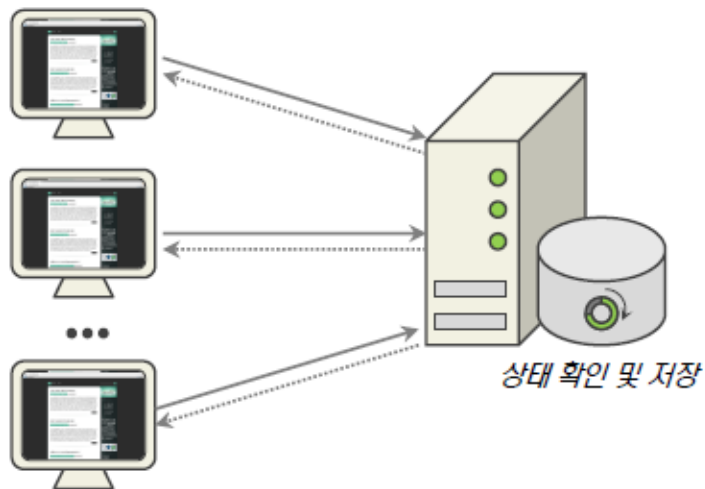




## 2.3 Stateless HTTP

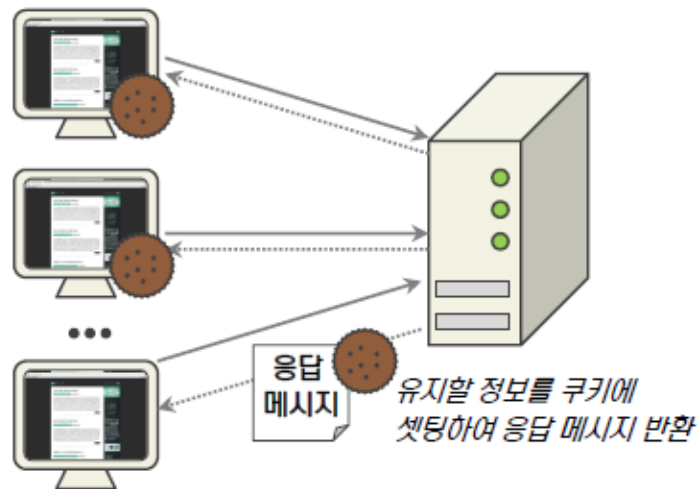
- ✓ HTTP는 기본적으로 상태를 유지하지 않는 Stateless 프로토콜입니다.
- ✓ Stateless 프로토콜은 과거 상태를 신경 쓰지 않기 때문에, 많은 양의 요청을 일관된 방법으로 빠르게 처리합니다.
- ✓ 반면 Stateful 프로토콜은 서버가 많은 클라이언트의 상태를 기억해야 하기 때문에 리소스에 부담을 줍니다.
- ✓ 또한 웹 애플리케이션에 쿠키(Cookie) 기술을 도입하면서, 사용자 식별 및 상태 유지 등 Stateless의 한계를 극복합니다.

상태를 유지하는(Stateful) 프로토콜



Stateful 프로토콜은 매번 요청이 일어날 때마다, 서버가 수많은 클라이언트의 상태를 유지하기 위해 상태 정보를 동기화하고 저장 하는 등의 단계를 거쳐야 합니다. 따라서 상태 유지는 서버의 CPU나 메모리에 부담을 주게 됩니다.

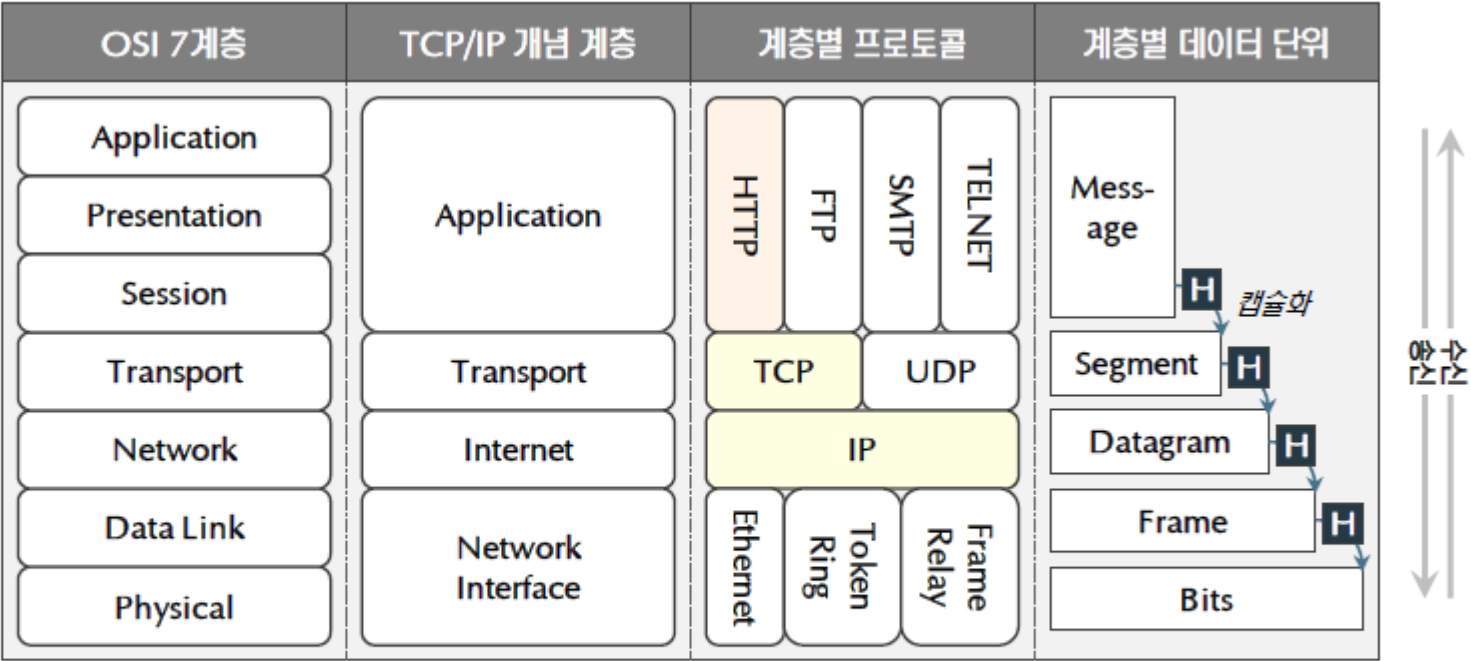
상태를 유지하지 않는(Stateless) 프로토콜



Stateless 프로토콜은 상태 유지를 위한 과정을 생략하기 때문에, 빠르게 요청을 처리합니다. 하지만 클라이언트를 식별하거나 과거 상태가 필요한 경우가 있습니다. 이 때 쿠키를 활용합니다. 서버가 기억할 정보를 쿠키에 담아 보내면 클라이언트는 이 쿠키를 보존합니다. 이 후 요청은 쿠키를 함께 보내기 때문에, 이를 통해 서버는 클라이언트의 상태를 알 수 있습니다.

## 2.4 TCP/IP와 HTTP (1/2)

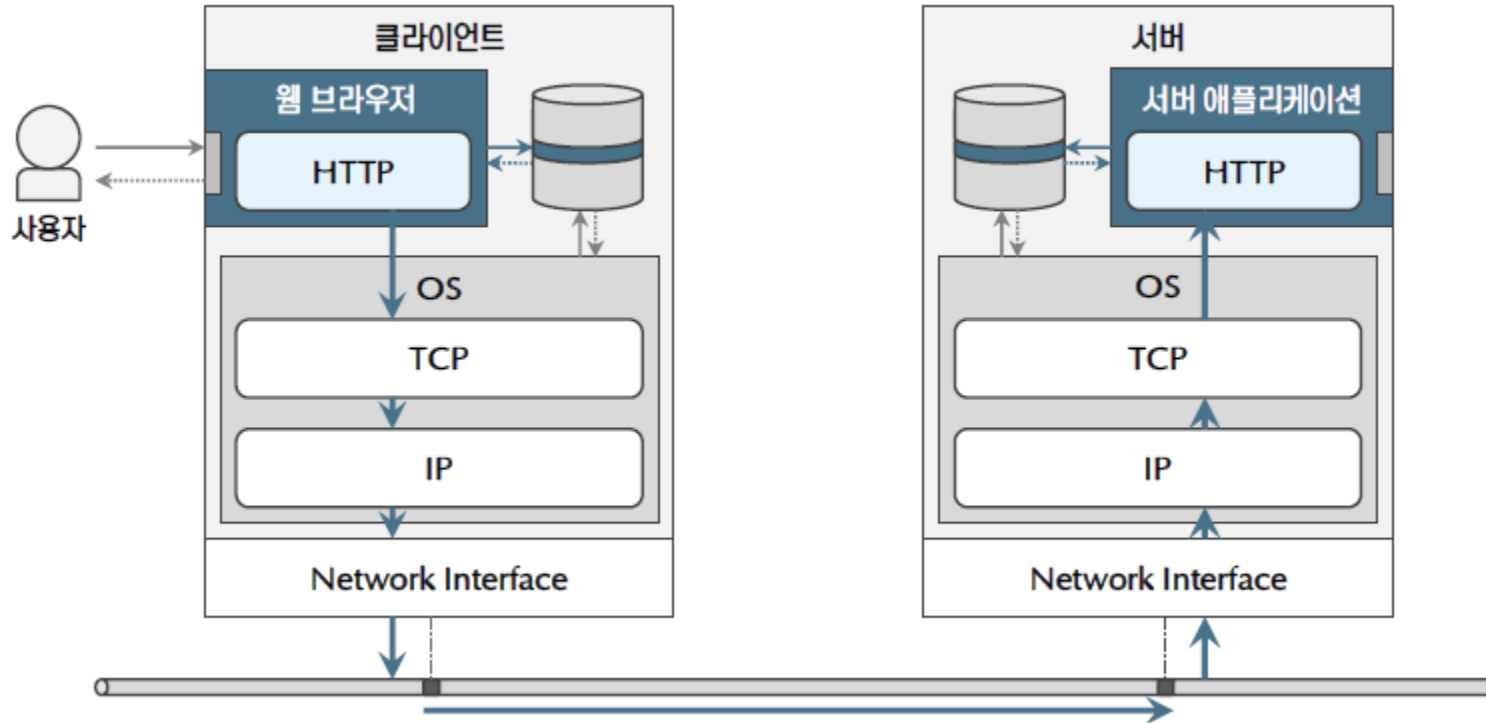
- ✓ TCP/IP(Transmission Control Protocol/Internetworking Protocol)는 컴퓨터의 네트워크 통신 규약입니다.
- ✓ TCP/IP는 총 다섯 계층으로 구성되며 각 계층별 네트워크마다 사용되는 프로토콜과 데이터 단위가 다릅니다.
- ✓ 각 계층에서 데이터를 전송할 때는 헤더로 감싸야 하며, 이 과정을 캡슐화(Encapsulation)이라고 합니다.
- ✓ TCP는 신뢰를 기반합니다. 데이터를 보내기 전 후로 상대의 상태를 확인하고 만약 실패했다면 재전송합니다.
- ✓ HTTP는 애플리케이션 계층에서 담당하는 부분만 고려하면 정확하게 메시지를 전송할 수 있습니다.



- OSI(Open System Interconnection, 개방형 시스템 상호접속)는 ISO(국제표준화기구)가 작성한 컴퓨터 통신절차에 관한 국제표준규격입니다.
- TCP/IP의 네트워크 연결 계층에서는 특정 프로토콜을 지정하기 보다는 모든 표준과 기술에 정의된 프로토콜을 지원합니다. 이더넷과 토큰링은 LAN, 프레임 릴레이는 WAN에 기반합니다.
- 애플리케이션 계층에는 FTP(File Transfer Protocol), SMTP(Simple Mail Transfer Protocol), POP(Post Office Protocol), Telnet(Tele Network) 등의 프로토콜이 있습니다.

## 2.4 TCP/IP와 HTTP (2/2)

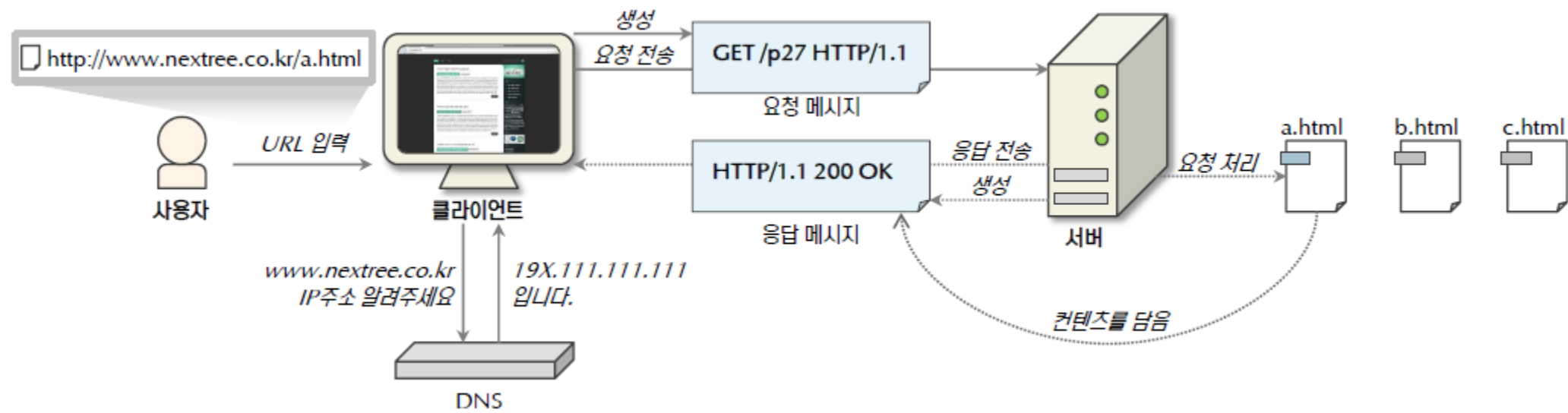
- ✓ 만약, TCP/IP가 계층 구조가 아니라면 어떤 네트워크 기술의 사양이 변경 되더라도 계층 전체가 변경되어야 합니다.
- ✓ 계층화하면 각 계층별로 독립적인 진화가 가능하며, 자신이 담당하는 기능만 수행하면 됩니다.
- ✓ HTTP는 애플리케이션에서 구현합니다. 클라이언트 역할을 하는 웹 브라우저와 서버 프로그램이 HTTP를 구현하고 있습니다.
- ✓ 애플리케이션은 OS에서 구현한 TCP/IP 서비스를 이용하여 통신하며, OS가 관리하는 저장소에서 자원을 관리합니다.





# 2.8 요약

- ✓ HTTP(Hyper-Text Transfer Protocol)은 하이퍼텍스트 문서를 전송하기 위한 통신 규약입니다.
- ✓ 클라이언트와 서버간의 통신을 위해 URL, 요청 메소드 등을 담은 요청 메시지와 상태코드를 담은 응답메시지를 구성합니다.
- ✓ HTTP는 기본적으로 상태를 유지하지 않는 Stateless 프로토콜입니다. 쿠키를 이용하면 상태를 유지할 수 있습니다.
- ✓ HTTP는 애플리케이션 계층 프로토콜로서 TCP와 통신합니다. TCP 덕분에 정확하게 데이터를 전송할 수 있습니다.





## 3. HTTP 이해

---

3.1 HTTP 메시지 구조

3.2 요청 메소드

3.3 URL

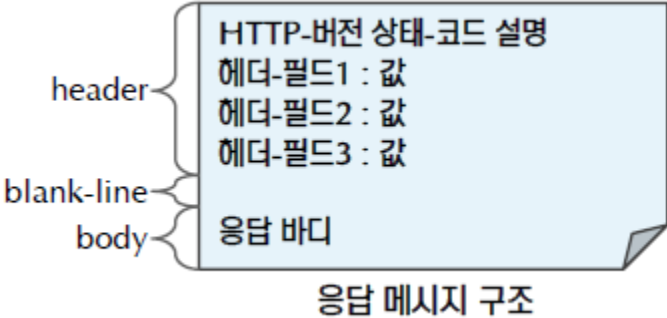
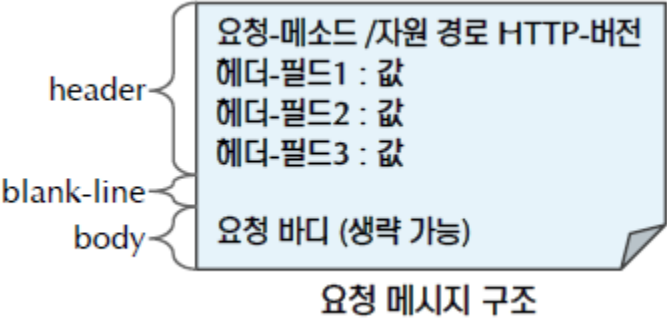
3.4 상태 코드

3.5 콘텐츠 타입

3.6 요약

# 3.1 HTTP 메시지 구조

- ✓ HTTP가 전달하는 정보는 반드시 HTTP 메시지 규격에 맞아야 합니다.
- ✓ HTTP 메시지는 요청 메시지와 응답 메시지로 구분합니다. 각 메시지들은 헤더(Header)와 바디(Body)로 구성됩니다.
- ✓ 헤더와 바디를 구분하기 위해 사이에 한 줄이 비어 있습니다. 이 줄을 CRLF(Carriage Return/Line Feed)라 부릅니다.
- ✓ 메시지 헤더의 첫 줄은 요청과 응답에 따라 서로 상이한 구조를 가집니다.
- ✓ 헤더에는 정보의 조건과 속성들을 담고 있으며 콜론(:)으로 구분하여 필드와 값을 나열합니다.





## 3.2 요청 메소드 (1/3) – 정의

- ✓ 클라이언트가 서버에게 보내는 요청 메시지의 헤더에는 요청 메소드(Request Method)가 있습니다.
- ✓ 자원의 경로가 같은 두 요청 메시지가 있더라도, 요청 메소드가 다르게 명시되어 있다면 이는 다른 요청입니다.
- ✓ 서버는 요청 메소드에 따라 다른 처리를 할 수 있습니다. 가장 많이 사용하는 요청 메소드는 GET과 POST입니다.
- ✓ 요청 메소드는 각각 다른 목적을 가집니다.

요청 메소드가 다른 요청 메시지

GET /posts/27 HTTP/1.1  
...

요청 메시지 1

≠

POST /posts/27 HTTP/1.1  
...

요청 메시지 2

위 그림의 두 요청 메시지는 /posts/27이라는 동일한 경로의 자원을 요청하고 있습니다. 그러나 두 메시지는 요청 메소드가 다르기 때문에 서버에서는 이 두 요청을 다르게 처리합니다. 서버는 같은 URL이더라도 요청 메소드 별로 다른 로직을 구현할 수 있으며, 구현하지 않은 요청 메소드에 대해서는 처리할 수 없습니다. 만약 구현 시 어떤 설정도 하지 않았을 경우의 디폴트 요청 메소드는 GET 방식입니다.

HTTP 요청 메소드의 종류	
GET	자원 취득
POST	정보 전송
PUT	자원 전송
DELETE	자원 삭제
OPTIONS	해당 URL로 지원하는 요청 메소드 종류 취득
HEAD	메시지 헤더 취득
TRACE	경로 조사

## 3.2 요청 메소드 (2/3) – GET vs POST

- ✓ GET과 POST를 구분해서 사용하는 가장 큰 이유는 질의 문자열이 작성되는 위치가 다르기 때문입니다.
- ✓ GET 요청은 파라미터의 이름과 값의 쌍을 URL 뒤에 물음표(?)로 구분하여 나열합니다.
- ✓ POST 요청은 파라미터를 메시지 바디에 작성하여 보내기 때문에 URL에 파라미터가 들어나지 않습니다.

### GET 방식의 질의 문자열

**GET** /board/posts?createDate=20160516&author=eykim

...(헤더필드)...

### POST 방식의 질의 문자열

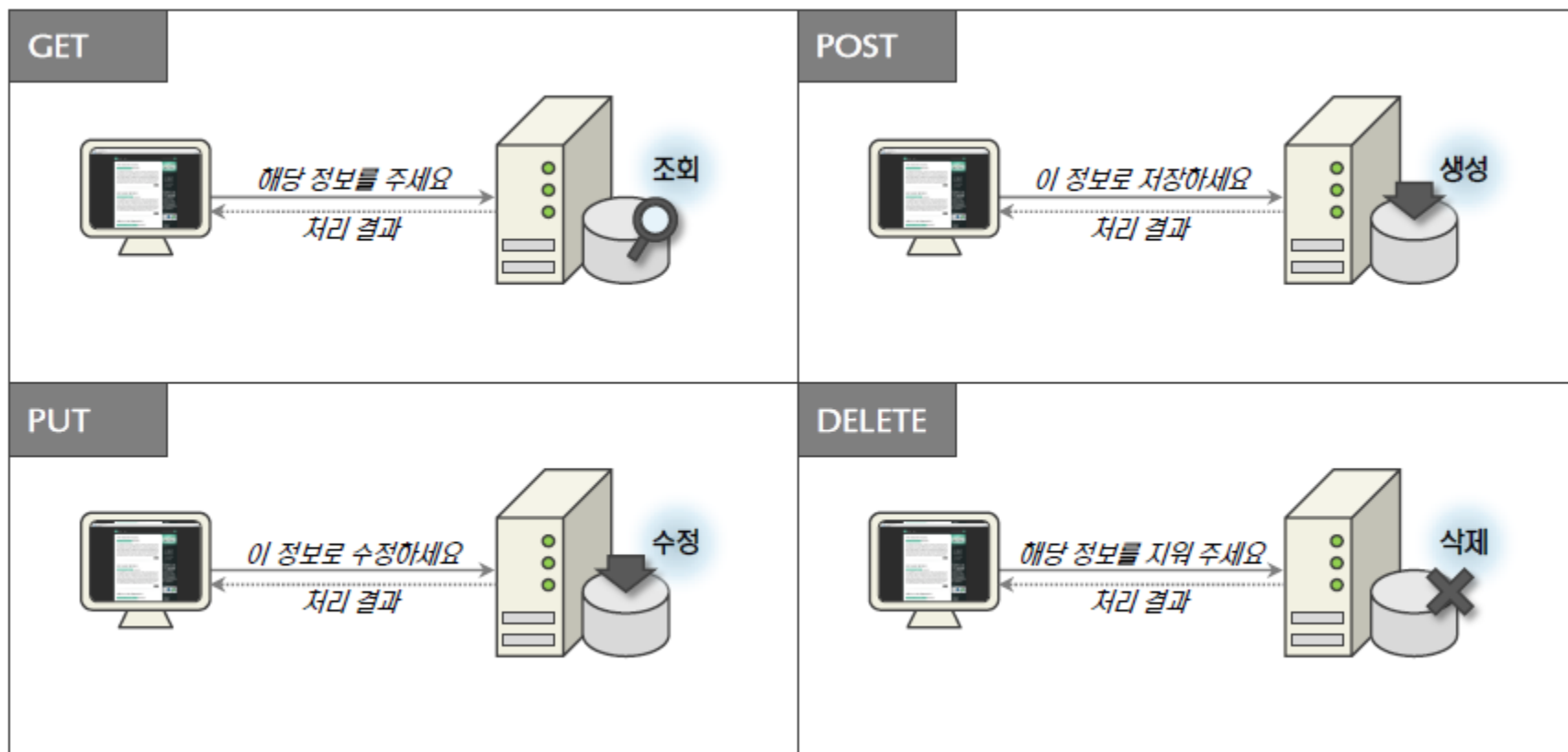
**POST** /board/posts

...(헤더필드)...

**createDate=20160516&author=eykim**

### 3.3 요청 메소드 (3/3) – RESTful 설계

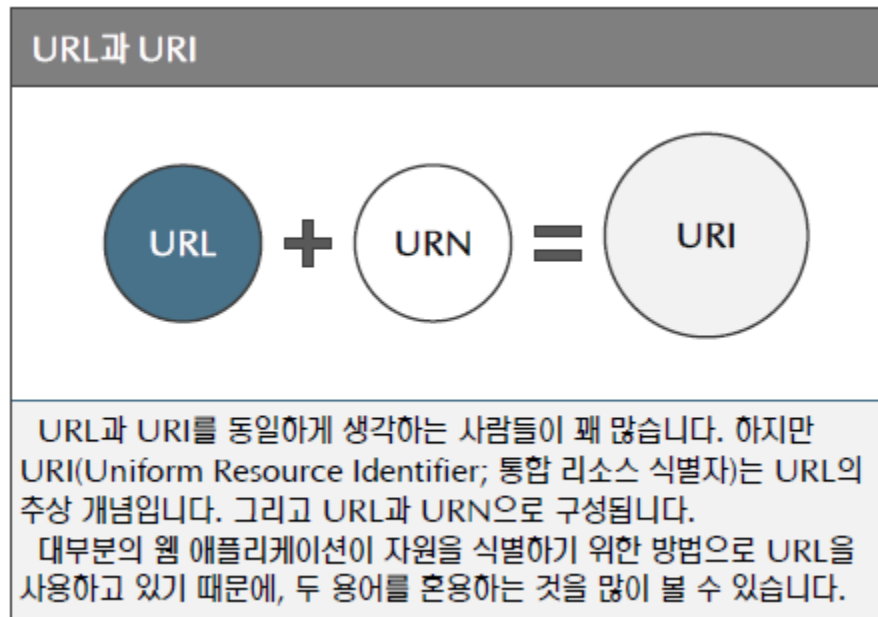
- ✓ 오늘 날의 웹 애플리케이션은 단순한 콘텐츠 교환에서 나아가, 수 많은 정보를 관리(생성/조회/수정/삭제)합니다.
- ✓ 클라이언트가 요청하는 자원은 물리적인 파일만을 의미하는 것이 아닙니다. 비즈니스에 활용되는 정보일 수도 있습니다.
- ✓ 따라서 요청 메소드는 클라이언트가 건네는 정보를 서버가 어떻게 활용할 지 결정하는 기준이 됩니다.
- ✓ REST(Representational State Transfer) 웹 서비스는 기능의 성격에 따라 요청 메소드를 설계합니다.





### 3.3 URL(1/3) – 정의

- ✓ URL(Uniform Resource Locator)은 자원을 탐색하기 위한 주소 체계 중 하나입니다.
- ✓ 서버는 요청 메시지에 적힌 URL을 확인하여 자원에 접근합니다.
- ✓ URL이 등장하기 전에는 파일을 공유하기 위해 사용자가 직접 서버에 접속해서 디렉토리들을 검색해야 했습니다.
- ✓ URL은 모든 자원들이 가지는 고유의 주소입니다. 따라서 URL을 통해 보다 쉽게 자원을 탐색할 수 있습니다.



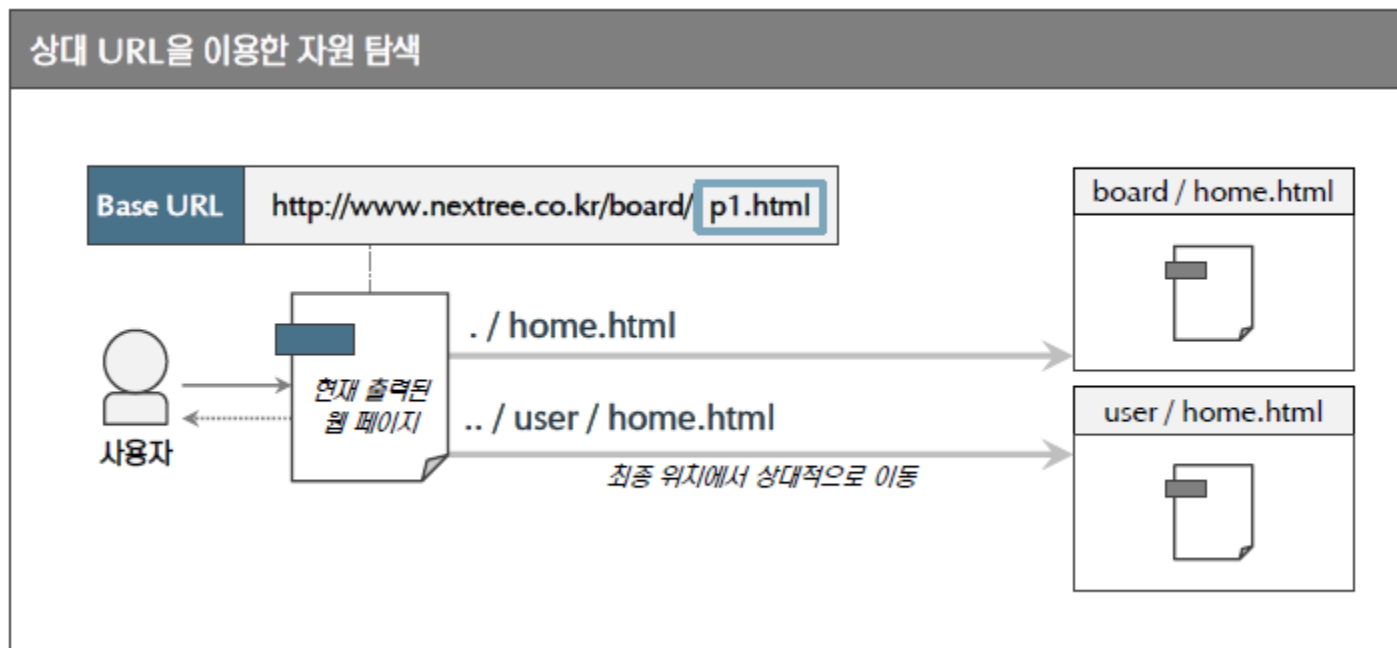
### 3.3 URL(2/3) – 문법

- ✓ http는 URL의 시킴입니다. 스킴은 자원에 어떤 프로토콜 혹은 애플리케이션을 이용하여 접근할지 결정합니다.
- ✓ 호스트는 서버의 이름이나 IP주소입니다. 호스트가 서버의 이름일 경우 DNS를 통해 IP주소를 찾아 접근합니다.
- ✓ 경로는 요청하는 리소스가 서버의 어느 위치에 있는지를 나타냅니다. 앞에 / 구분자를 입력하여 나열합니다.

스킴 :// 호스트 : 포트 / 경로 ? 질의 문자열 # 프래그먼트									
상세 설명									
스킴	리소스에 어떤 프로토콜을 이용하여 접근할 것인지 결정하고, 접근하기 위한 애플리케이션을 결정합니다. http, ftp 등 프로토콜 연결을 위한 스킴과, mailto, tel 등 애플리케이션 연결을 위한 스킴이 있습니다.								
호스트	리소스를 연결할 서버의 이름이나 IP 주소를 가리킵니다. IP주소가 아닌 서버의 이름으로 입력했을 경우, 서버의 정확한 위치를 알기 위해 DNS(Domain Name Service)를 이용합니다.								
포트	TCP 포트는 서버(하드웨어) 에서 구동되는 소프트웨어(서비스)를 구별하기 위한 번호입니다. (0~65535) 서버는 포트를 통해 클라이언트가 어느 애플리케이션에 접속하기를 원하는지 알 수 있습니다. HTTP 스킴 일 경우 포트를 생략하면 80번 포트를 사용합니다.								
경로	URL경로는 서버의 어디에 리소스가 있는지 가리킵니다. / 구분자를 이용하여 파일시스템에 접근하듯이 작성합니다. 웹 애플리케이션에서 서버가 운영중인 모듈을 가리키는 경로를 Context Path라고 합니다.								
질의 문자열	? 뒤에 나오는 문자열은 질의 문자열 (Query String)이라고 부릅니다. 주로 클라이언트에서 서버로 요청하는 조건을 나타냅니다. 여러 조건을 나열할 때는 &로 구분하여 나열합니다. (?no=1&name=ey)								
프래그먼트	프래그먼트는 서버가 응답한 자원의 일부분을 가리킵니다. 일반적으로 HTML의 Frame, div 등 영역을 나타내는 태그의 id를 의미합니다. 프래그먼트는 서버로 보내지 않고 브라우저가 직접 해당 프래그먼트가 있는 곳으로 스크롤을 내려 출력합니다.								

### 3.3 URL(3/3) – 상대 URL vs 절대 URL

- ✓ URL은 상대 URL과 절대 URL로 나뉩니다. 절대 URL은 스킴부터 경로까지 모든 정보를 가지고 있습니다.
- ✓ 상대 URL은 현재 출력되고 있는 자원의 위치(Base URL)를 기준으로 상대적인 위치를 가리킵니다.
- ✓ 상대 URL은 URL의 일부이기 때문에 브라우저 주소 창이 아닌 서로 다른 하이퍼텍스트를 연결할 때 사용됩니다.
- ✓ 서버의 IP는 개발/테스트/운영 환경에 따라 변경될 가능성이 크므로 상대 URL을 이용하면 변경을 줄일 수 있습니다.



# 3.4 상태 코드

- ✓ 상태 코드(Status Code)는 클라이언트가 보낸 요청이 서버에서 어떻게 처리되었는지 알려 주는 역할을 합니다.
- ✓ 요청을 처리하면서 나타날 수 있는 다양한 결과들을 코드화하여 클라이언트가 상태 코드만 봐도 결과를 예측할 수 있습니다.
- ✓ 상태 코드는 요청이 정상 처리 되지 않았을 때 더욱 중요한 역할을 합니다.
- ✓ 상태 코드를 인지하고 있는 웹 애플리케이션 개발자는 오류가 발생했을 때 더욱 신속하게 해결할 수 있습니다.

상태 코드 분류		대표 상태코드	
1XX	정보성 (연결 여부, 적용 여부)	200 OK	정상 처리 상태 코드 입니다.
2XX	성공	301 Moved Permanently	해당 URL에 연결된 자원이 없음을 의미합니다.
3XX	리다이렉트	400 Bad Request	클라이언트가 형식에 맞지 않는 요청을 보냈을 때 나타납니다.
4XX	클라이언트 오류	404 Not Found	301 ~ 307은 요청이 다른 URL로 리다이렉트 되었음을 알립니다.
5XX	서버 로직상의 오류	405 Method Not Allowed	지원하지 않는 메소드를 요청하였을 경우 발생합니다.
		500 Internal Server Error	주로 동적 처리 중 서버 로직에서 오류가 발생하였음을 나타냅니다.

# 3.5 콘텐츠 타입 (1/3) – 정의

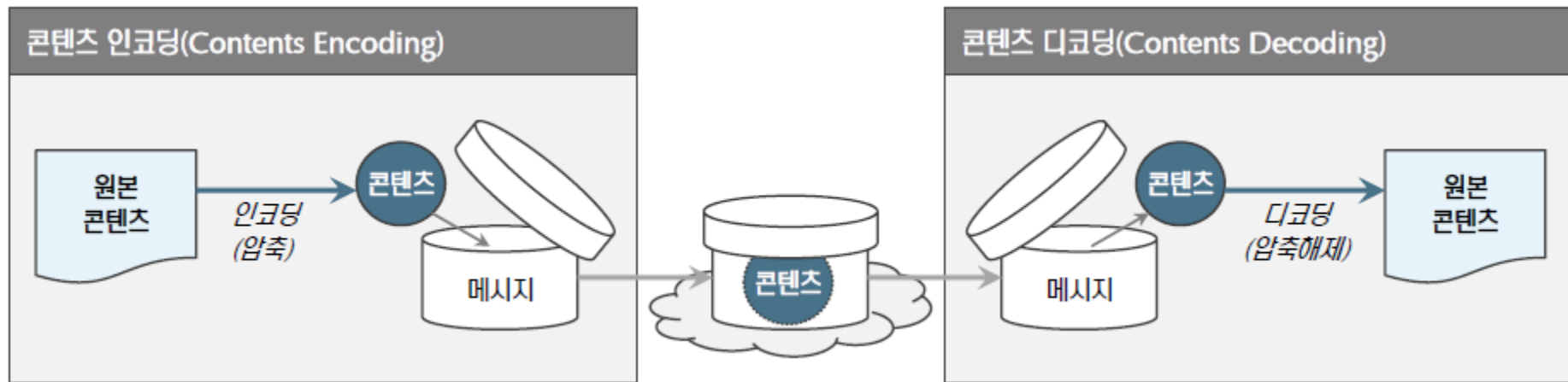
- ✓ HTTP는 매일 수많은 콘텐츠를 전송합니다. 또한 모든 콘텐츠는 HTTP 메시지에 담겨 전송됩니다.
- ✓ 메시지 내 콘텐츠가 어떤 유형인지 알리기 위해, 헤더에 콘텐츠 타입(헤더 필드 Content-type)을 명시합니다.
- ✓ 콘텐츠 타입의 값은 MIME(Multipurpose Internet Mail Extensions)형식의 주 타입/부 타입으로 나타냅니다.
- ✓ 클라이언트와 서버는 헤더에 명시된 콘텐츠 타입을 보고 엔티티(메시지 바디 본문)를 해석하여 처리합니다.

대표 MIME 콘텐츠 타입	
text/html	HTML 문서를 나타내며, 가장 많이 사용하는 콘텐츠 타입입니다.
text/plain	순수 Text를 나타내며, 간단한 AJAX 통신에서 주로 등장합니다.
text/xml	XML 문서를 나타내며, AJAX 통신에서 주로 등장합니다.
text/json	json 객체를 나타내며, AJAX 통신에서 주로 등장합니다.
image/jpeg	jpeg, 혹은 jpg 확장자를 가지는 콘텐츠를 나타냅니다.
image/png	png 확장자의 이미지 콘텐츠를 나타냅니다.



## 3.5 콘텐츠 타입 (2/3) – 인코딩

- ✓ 대부분의 HTTP 서버는 메시지의 전송 시간을 단축시키기 위해 콘텐츠를 압축(인코딩)하여 전송합니다.
- ✓ 클라이언트가 인코딩(압축)된 콘텐츠를 받으면, 디코딩(압축을 해제)하여 콘텐츠 원본을 출력합니다.
- ✓ 클라이언트는 헤더 속성 Accept-Encoding에 자신이 해석할 수 있는 인코딩 유형을 나열해 서버에게 알립니다.
- ✓ 서버는 헤더 속성 Content-encoding에 인코딩 타입을 명시하여, 클라이언트가 해당 유형으로 처리하도록 합니다.
- ✓ 가장 대표적인 콘텐츠 인코딩 유형에는 gzip이 있습니다.



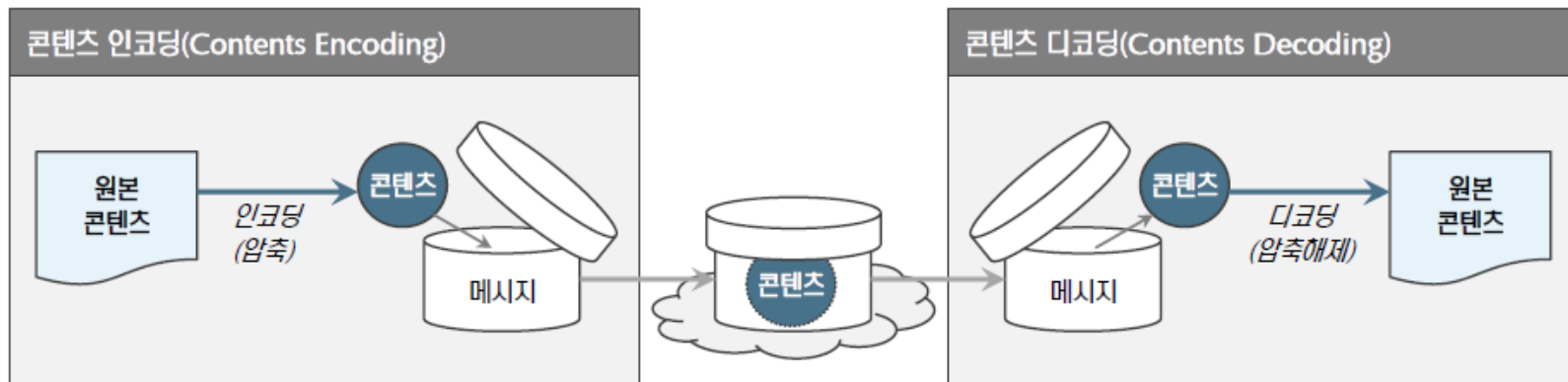
### 3.5 컨텐츠 타입 (3/3) – Charset

- ✓ 대부분의 개발자들은 컨텐츠 인코딩보다도 문자 집합 인코딩 때문에 많은 어려움을 겪습니다.
- ✓ 컨텐츠 타입을 지정할 때 MIME 타입 우측에 기재하는 차셋(charset)의 값이 바로 문자 인코딩 유형을 나타냅니다.
- ✓ 메시지를 보낼 때 컨텐츠의 문자를 비트로 변환하여 전송하고, 메시지를 받아 이 비트들을 다시 문자로 변환합니다.
- ✓ 이 때 서로 변환의 기준이 되는 차셋(charset) 값이 다르다면, 원하지 않은 문자로 깨져서 화면에 출력됩니다.

대표 MIME 차셋 값	
iso-8859-1	HTML의 기본 데이터 셋입니다. 네덜란드, 노르웨이, 덴마크, 프랑스, 영국, 이탈리아 그리고 몇 개의 아프리카어를 지원합니다. 유럽 국가의 언어를 지원하는 인코딩이라고 볼 수 있습니다.
euc-kr	한글을 지원하는 인코딩입니다. 아스키와 거의 비슷하고 역슬래시와 원화 기호가 추가된 것만 다릅니다. 하지만 11172자의 한글 중 2350자만을 지원하여 "똥,함,똥" 등의 글자는 표현하지 못합니다.
ms949	마이크로소프트사에서 euc-kr을 확장하여 제공하는 문자 집합입니다. euc-kr 보다는 많은 한글을 표현할 수 있으나 UTF-8보다는 사용이 불편화되지 못했습니다.
utf-8	가장 보편적인 문자 인코딩입니다. 유니코드로 글자를 변환하고 대부분의 언어를 모두 호환할 수 있습니다. 가변 길이의 코드로 문자를 지원하기 때문에 많은 글자를 효율적으로 표현 가능합니다.

## 3.6 요약

- ✓ HTTP는 프로토콜 규격에 맞는 요청-응답 메시지를 작성합니다.
- ✓ 요청 메시지에는 자원 처리 방식에 대한 요청 메소드와 자원의 위치를 가리키는 URL을 포함합니다.
- ✓ 응답 메시지에는 자원의 처리 결과에 관한 상태 코드를 포함합니다. 또한 반환하는 콘텐츠의 타입을 헤더에 명시합니다.
- ✓ 메시지 바디에는 콘텐츠가 담겨 있습니다. 인코딩을 통한 전송으로 전송 속도를 높일 수 있습니다.





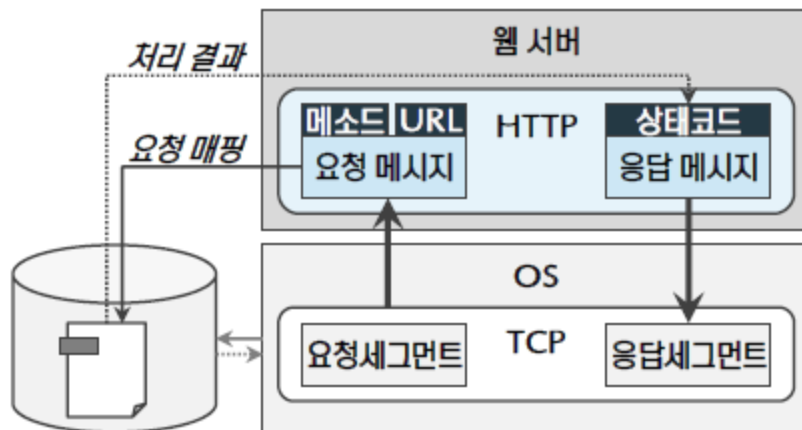
## 4. 웹 서버

---

- 4.1 웹 서버의 역할
- 4.2 가상 호스팅
- 4.3 프록시
- 4.4 서버 확장
- 4.5 보안
- 4.6 요약

## 4.1 웹 서버의 역할 (1/2) - 개요

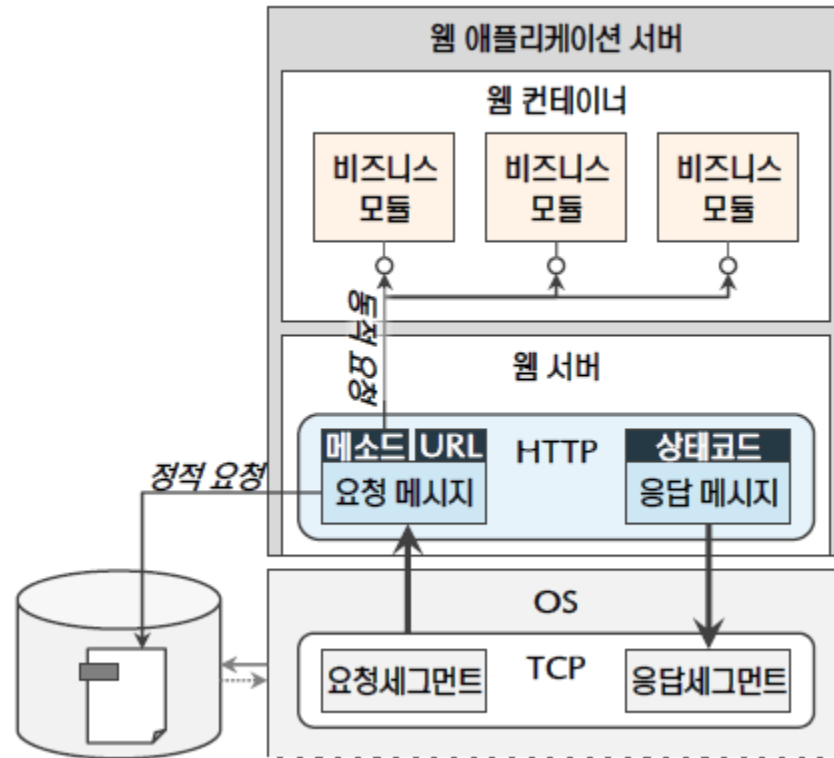
- ✓ 웹 서버는 클라이언트에게 받은 요청에 대한 자원을 찾아 응답하는 역할을 가지는 애플리케이션입니다.
- ✓ HTTP 프로토콜 스펙을 구현하고 있기 때문에 HTTP 요청 메시지를 분석하고 HTTP 응답 메시지를 생성할 수 있습니다.
- ✓ 또한 요청 메시지의 요청 메소드와 URL로 자원을 매핑하고, 처리 결과를 상태 코드로 변환합니다.
- ✓ 웹 서버는 TCP와 커넥션을 맺고 HTTP 메시지를 TCP의 데이터 규격인 세그먼트로 변환하여 데이터를 송수신 합니다.





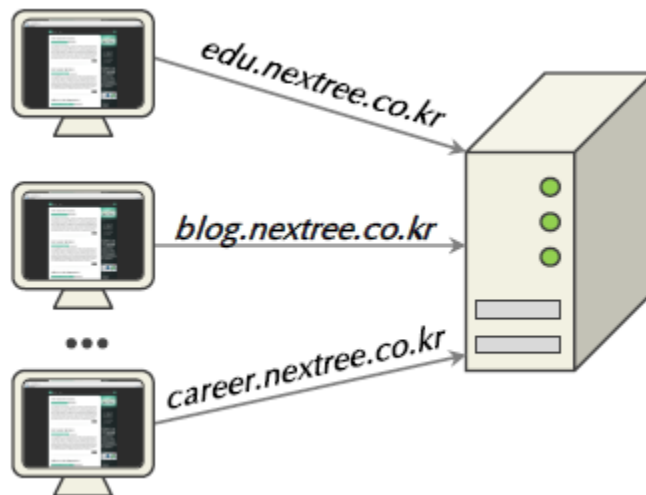
## 4.1 웹 서버의 역할 (2/2) – 웹 애플리케이션 서버

- ✓ 초기 웹 시스템은 콘텐츠를 교환하는 정도의 정적인(Static) 요청만 처리할 수 있었습니다.
- ✓ 이러한 웹 서버의 한계를 극복하기 위해 웹 애플리케이션 서버(WAS)가 등장합니다.
- ✓ 웹 애플리케이션 서버는 클라이언트의 요청을 물리적인 파일 뿐 아니라 기능을 수행하는 서비스와 연결할 수 있습니다.
- ✓ 따라서, 웹 애플리케이션 서버의 등장으로 대규모의 웹 시스템들을 개발할 수 있습니다.



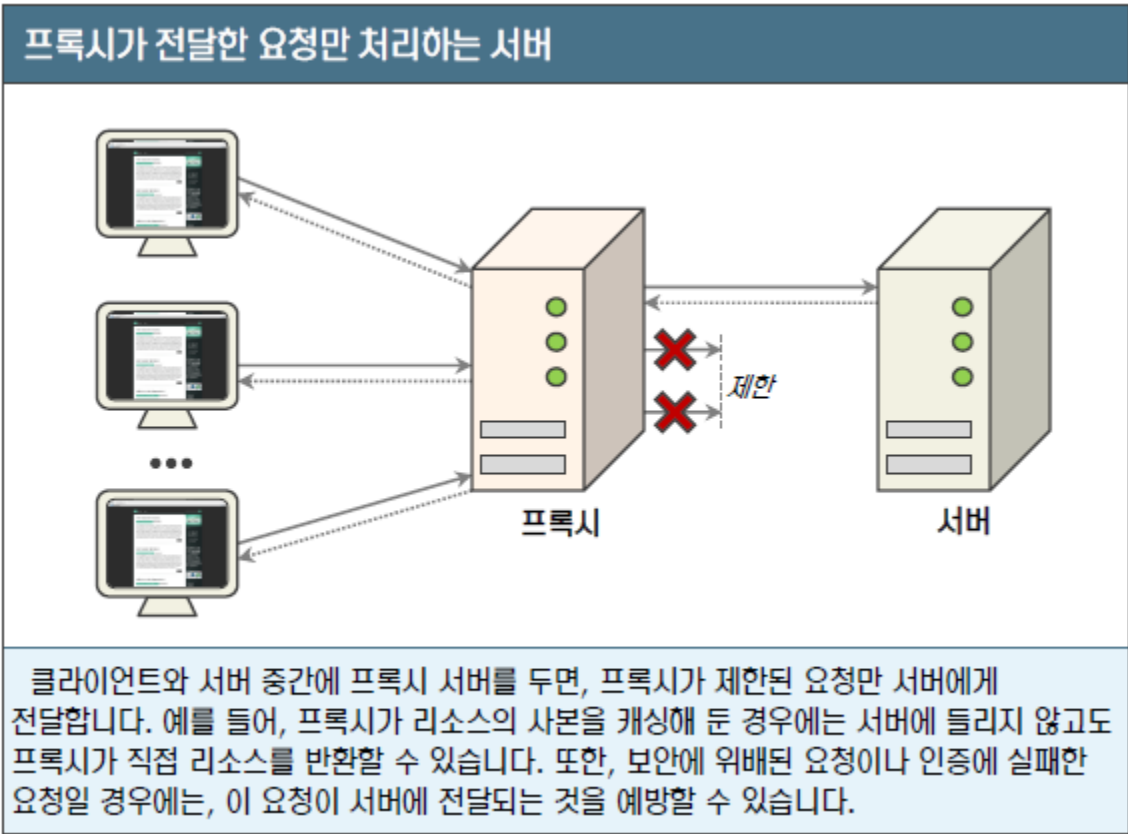
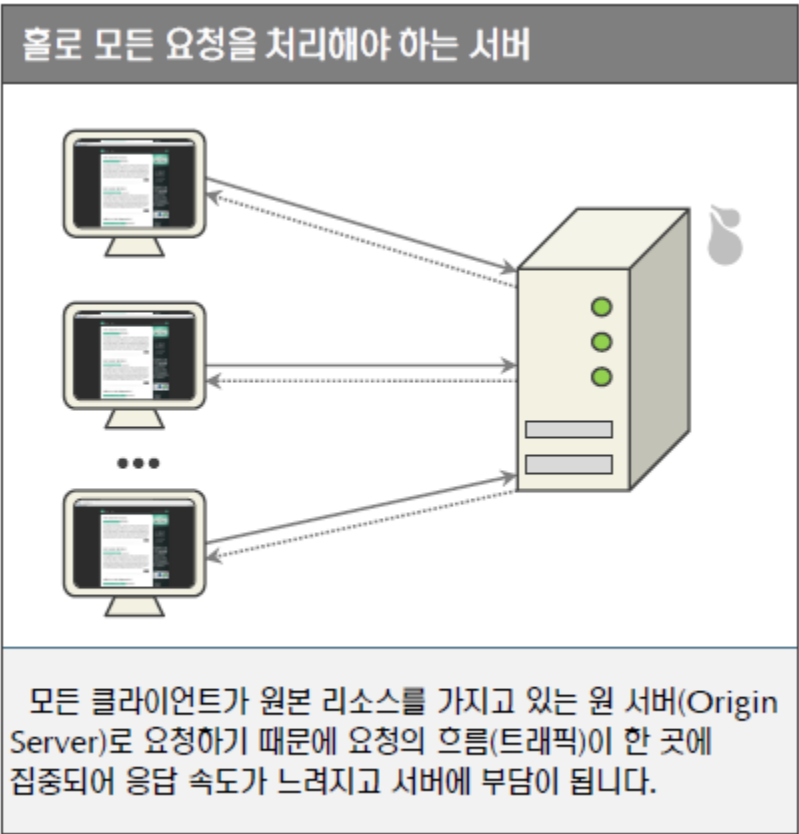
## 4.2 가상 호스팅

- ✓ 호스트는 서버의 이름 혹은 별칭입니다. 클라이언트는 호스트를 이용하여 서버에 접근할 수 있습니다.
- ✓ 가상 호스팅(Virtual Hosting)은 한 대의 서버에 여러 호스트를 매핑합니다.
- ✓ 웹 시스템이 장비를 따로 구축할 규모가 아니라면, 가상 호스팅을 이용해 서버 구축 및 운영 비용을 절감하는 것이 좋습니다.
- ✓ 이 때 DNS에 등록된 호스트와 서버에 추가한 호스트 이름이 동일해야만 가상 호스팅을 할 수 있습니다.



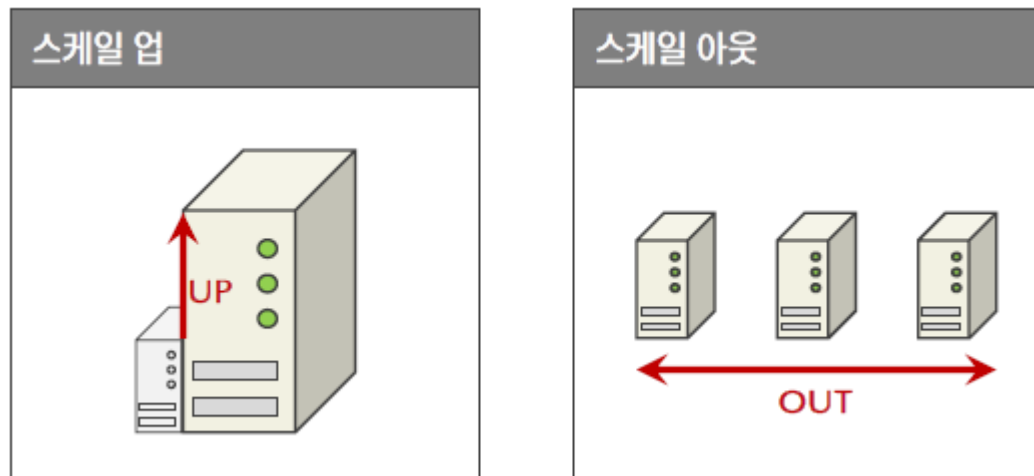
## 4.3 프록시

- ✓ 프록시(Proxy)는 클라이언트와 서버의 중개인(Broker) 역할을 합니다.
- ✓ 프록시는 서버로 가는 요청을 제어할 수 있기 때문에, 웹 캐시, 보안 방화벽, 서비스 접근 제어자 등 다양하게 활용됩니다.
- ✓ 원 서버가 홀로 모든 요청을 검증하고 처리하기에는 부담이 있기 때문에 프록시를 이용하여 서버의 트래픽을 절감합니다.
- ✓ 프록시는 자신이 처리할 수 있는 요청이 있으면 서버까지 전달하지 않고 바로 응답하기 때문에 응답 속도를 높입니다.



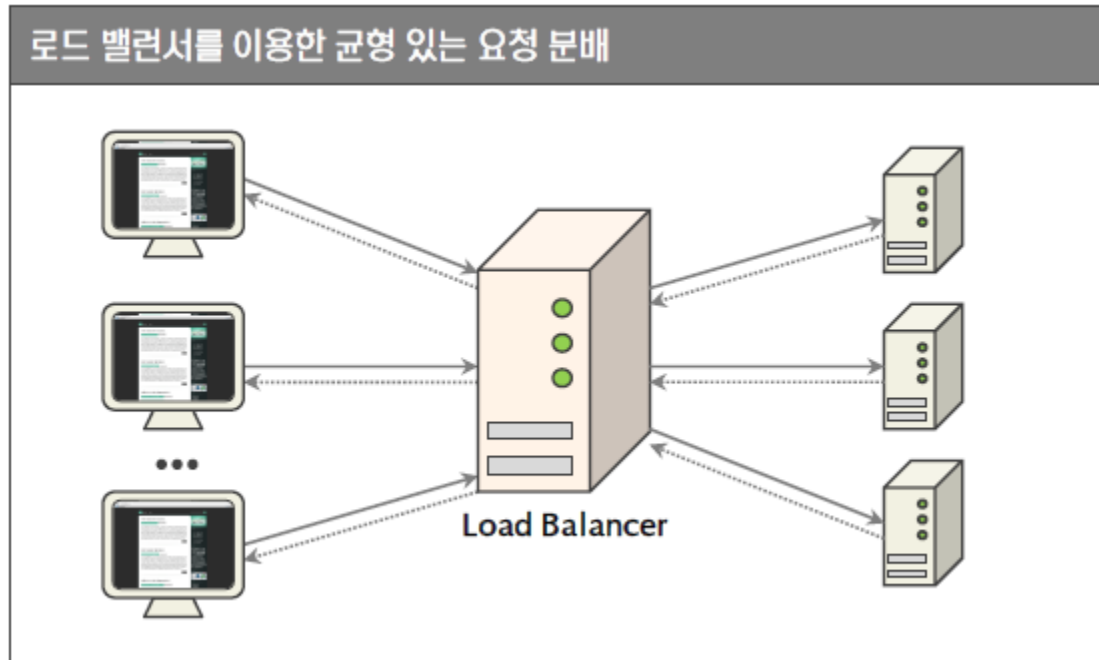
## 4.4 서버 확장 (1/3) – Scale-UP vs Scale-Out

- ✓ 프록시를 적용해도 원본 리소스는 하나의 서버가 처리하기 때문에 여전히 트래픽 부하에 대한 문제가 남아 있습니다.
- ✓ 서버의 성능을 개선하는 방법에는 스케일-업(Scale-Up)과 스케일-아웃(Scale-Out) 두 가지가 있습니다.
- ✓ 스케일-업은 서버 장비의 사양을 높여서 처리 능력을 높이는 방법입니다. 하지만 여전히 한 대의 서버로 부하가 집중됩니다.
- ✓ 스케일-아웃은 서버의 대수를 늘리는 것입니다. 여러 대의 서버가 요청을 분담하여 처리하여 부하를 줄일 수 있습니다.



## 4.4 서버 확장 (2/3) – 로드 밸런싱

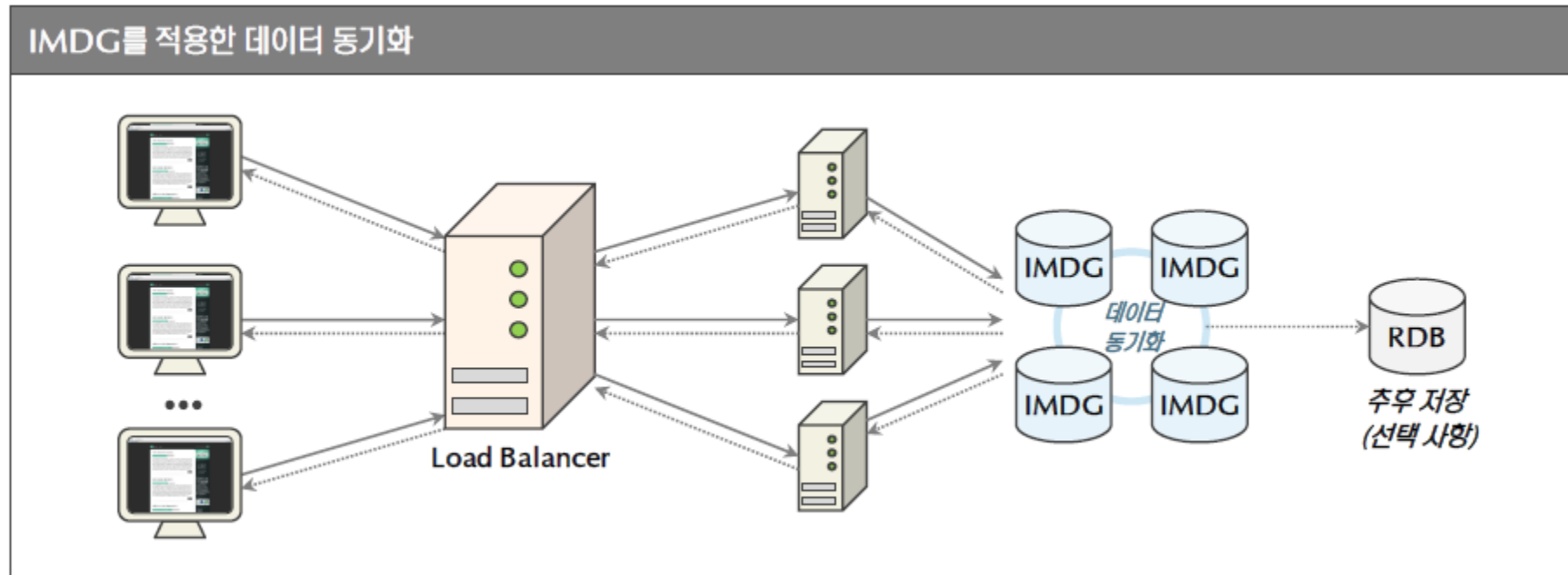
- ✓ 스케일 아웃을 통한 서버 확장은 여러 대의 서버가 분담하여 일을 처리하기 때문에 서버의 부하를 줄일 수 있습니다.
- ✓ 그러나 여러 대의 서버가 균형 있게 일을 분배 받지 못한다면 여전히 서버 부하의 문제점을 해결될 수 없습니다.
- ✓ 서버의 트래픽이 어느 정도 집중되어 있는지 파악하고 여유가 있는 서버에게 요청을 전달하는 것이 바로 로드 밸런싱입니다.
- ✓ 직접 구현하지 않아도 L4, L7, HAPROXY 등의 로드밸런서를 적용하면 쉽게 로드 밸런싱을 실현할 수 있습니다.





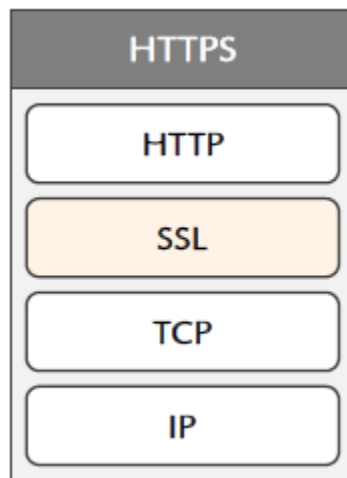
## 4.4 서버 확장 (3/3) – 데이터 동기화

- ✓ 스케일 아웃으로 서버를 확장했을 때 로드 밸런싱 외에도 주의해야 할 사항이 있습니다. 바로 동기화된 데이터 제공입니다.
- ✓ 여러 서버가 동시에 요청을 처리하기 때문에, 각각 자원을 관리할 경우 서로 다른 상태의 자원을 반환할 수 있습니다.
- ✓ 하지만 동기화를 위해 여러 서버가 동일한 데이터 서버에 접근한다면 서버 부하 문제는 또 다시 발생합니다.
- ✓ HazelCast와 같은 IMDG(In Memory Data Grid)를 활용하여 빠르고 동기화된 데이터를 제공할 수 있습니다.
- ✓ IMDG는 메모리에 오브젝트를 그대로 매핑하여 속도가 빠르고, 주기적으로 데이터를 동기화하므로 신뢰도가 높습니다.



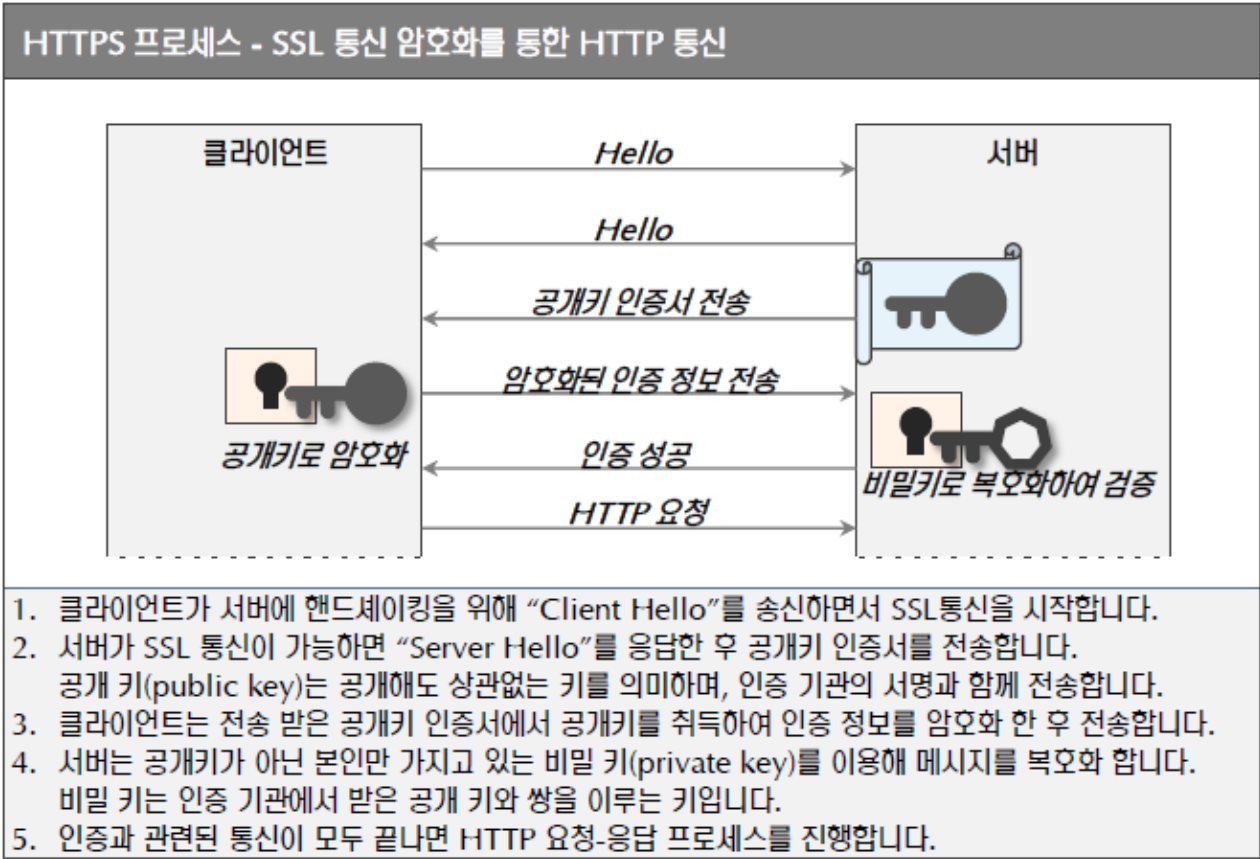
## 4.5 보안 (1/2) – 개요

- ✓ HTTP는 메시지로 데이터를 주고 받기 때문에 보안에 취약합니다.
- ✓ 암호화를 통해 데이터를 안전하게 전송할 수 있는데, 암호화는 콘텐츠 암호화와 통신 암호화 두 가지가 있습니다.
- ✓ 콘텐츠 암호화는 메시지에 암호화된 콘텐츠를 담는 것입니다. 하지만 메시지가 도착되면 복호화를 시도할 수 있습니다.
- ✓ 통신 암호화는 SSL(Secure Socket Layer)와 같은 암호화 프로토콜을 거쳐서 데이터를 보호하는 것입니다.
- ✓ 보안에 취약한 HTTP를 개선하기 위해 SSL을 적용시킨 구조를 바로 HTTPS 라고 부릅니다.



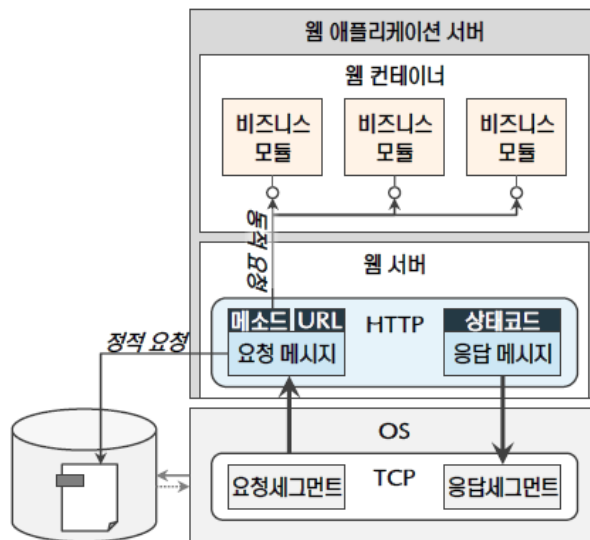
# 4.5 보안 (2/2) – HTTPS

- ✓ SSL을 사용한 HTTP를 HTTPS(HTTP Secure)이라고 합니다. HTTPS의 URL 스킴은 http://가 아닌 https://입니다.
- ✓ SSL은 공개키 암호화 방식을 사용합니다. 공개 키를 받아서 암호화하고 자신의 비밀 키를 이용해 복호화합니다.
- ✓ 우리 나라에서는 보편적으로 공인인증서라는 공개키 인증서를 사용하며, 대표적인 SSL 구현체로는 OpenSSL이 있습니다.
- ✓ HTTPS는 하나의 계층이 더 있기 때문에 통신 속도가 느리고 메모리나 CPU가 암호화를 수행하느라 처리가 느립니다.



## 4.6 요약

- ✓ 웹 서버는 클라이언트에게 받은 요청에 대한 자원을 찾아 응답하는 역할을 가지는 애플리케이션입니다.
- ✓ 웹 애플리케이션 서버는 클라이언트의 요청을 물리적인 파일 뿐 아니라 기능을 수행하는 서비스와 연결합니다.
- ✓ 프록시와 서버 확장을 통해 부하를 줄일 수 있습니다. 스케일-아웃 확장시에는 로드밸런싱과 데이터 동기화를 고려합니다.
- ✓ 보안이 중요한 웹 시스템에서는 통신 암호화 프로토콜 SSL과 HTTP를 연결하는 HTTPS를 적용합니다.





## 5. 웹 브라우저

---

5.1 웹 브라우저의 역할

5.2 캐싱

5.3 쿠키

5.4 브라우저 페이지

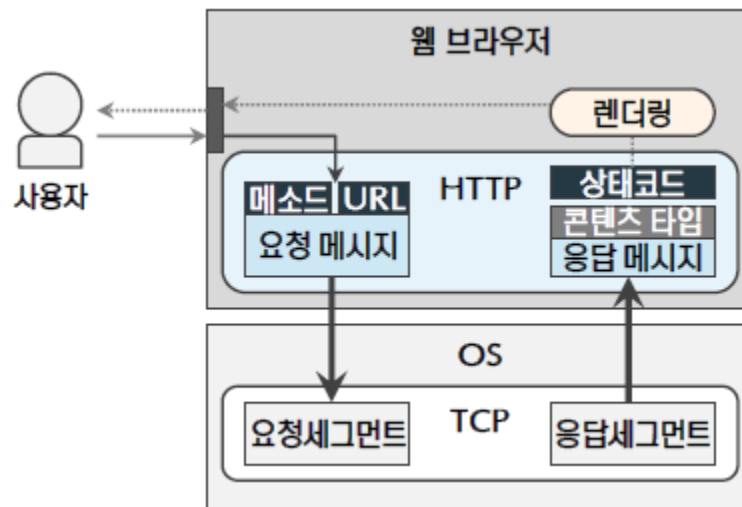
5.5 AJAX

5.6 요약



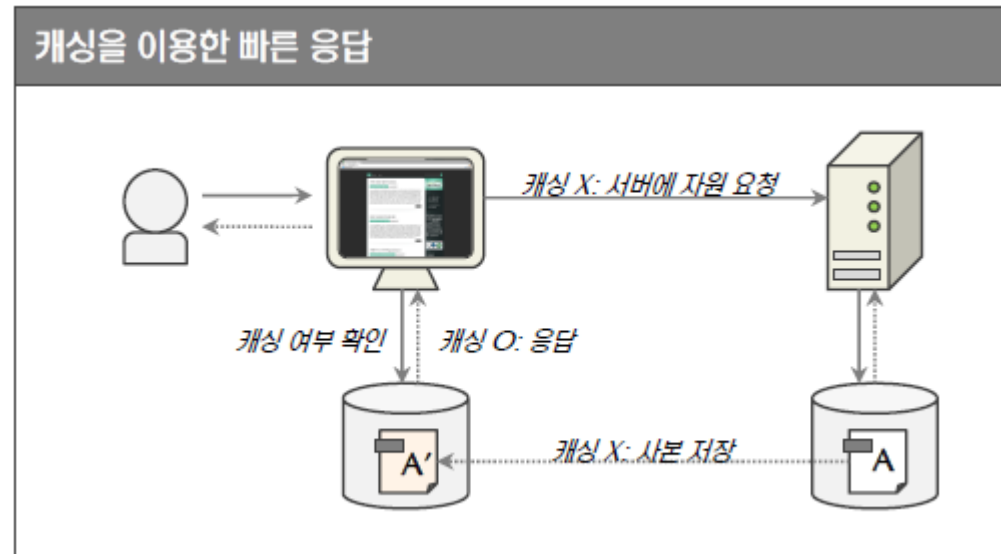
## 5.1 웹 브라우저의 역할

- ✓ 웹 브라우저는 HTTP 통신의 클라이언트 역할을 수행합니다. 즉, 요청을 보내고 응답을 받아 사용자에게 제공합니다.
- ✓ 사용자가 입력하거나 사용자의 이벤트와 연결된 URL과 요청 메소드를 기반으로 HTTP 규격의 요청 메시지를 생성합니다.
- ✓ 네트워크 계층을 따라 서버에게 요청을 보내고, 서버에서 부터 응답이 오면 응답 메시지를 분석합니다.
- ✓ 상태 코드가 정상이면 응답 메시지의 Content-Type에 맞게 콘텐츠를 렌더링 합니다.
- ✓ 그 밖에 서버에서 요청한 인증서 제공 및 쿠키에 저장한 상태 값 전송 그리고 캐싱을 통한 빠른 응답 등의 기능을 합니다.



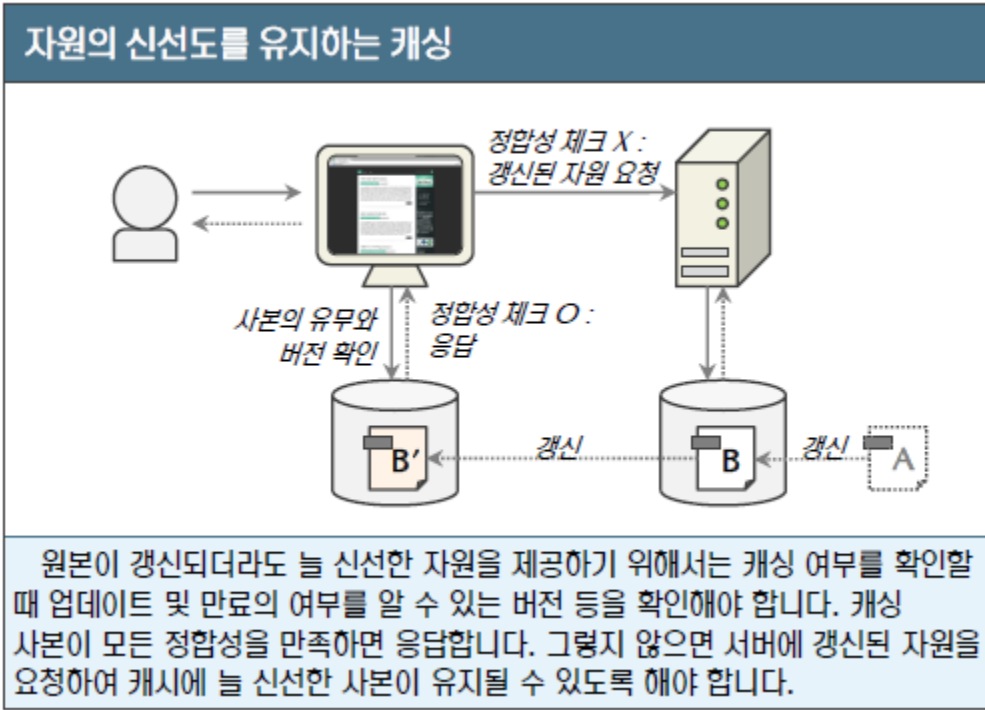
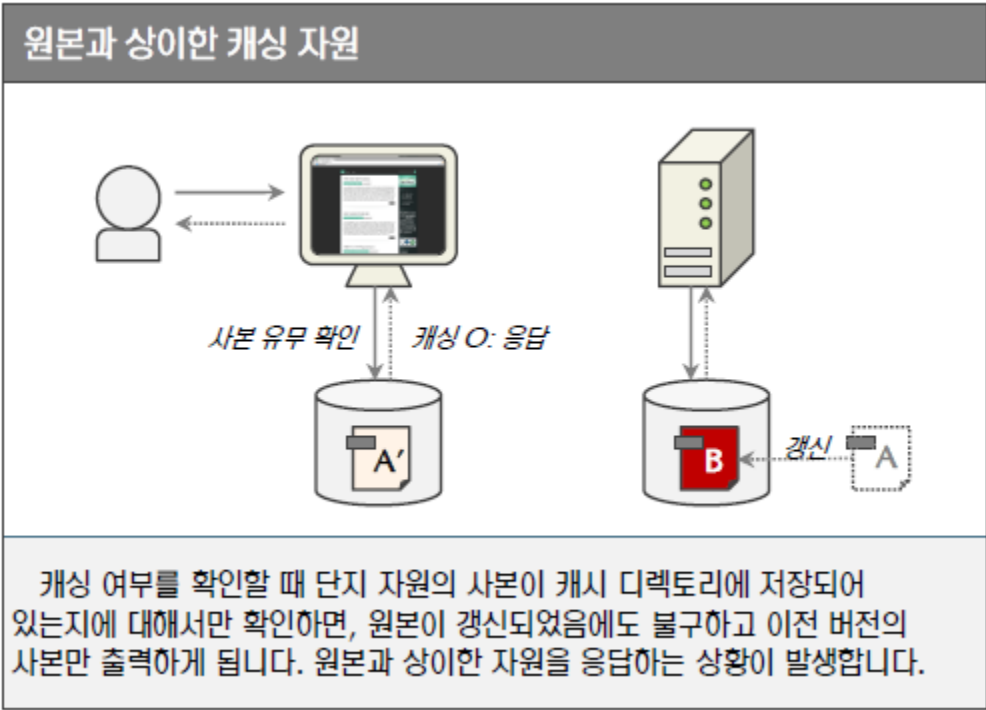
## 5.2 캐싱 (1/2) - 개요

- ✓ 캐싱은 요청을 한 클라이언트 또는 프록시 서버에 원본 리소스의 사본을 저장하여 빠르게 자원에 접근하는 것을 의미합니다.
- ✓ 한 번 요청한 자원에 대해서 사본을 저장해 놓으면 매 번 원본 리소스를 가진 원 서버까지 갈 필요가 없습니다.
- ✓ 캐싱은 원 서버의 부하를 줄이며 가까운 곳에서 자원을 찾아오기 때문에 응답 속도를 높입니다.
- ✓ 웹 브라우저들은 응답받은 콘텐츠들을 특정 디렉토리에서 관리합니다. 이 디렉토리를 브라우저 캐시라고 합니다.



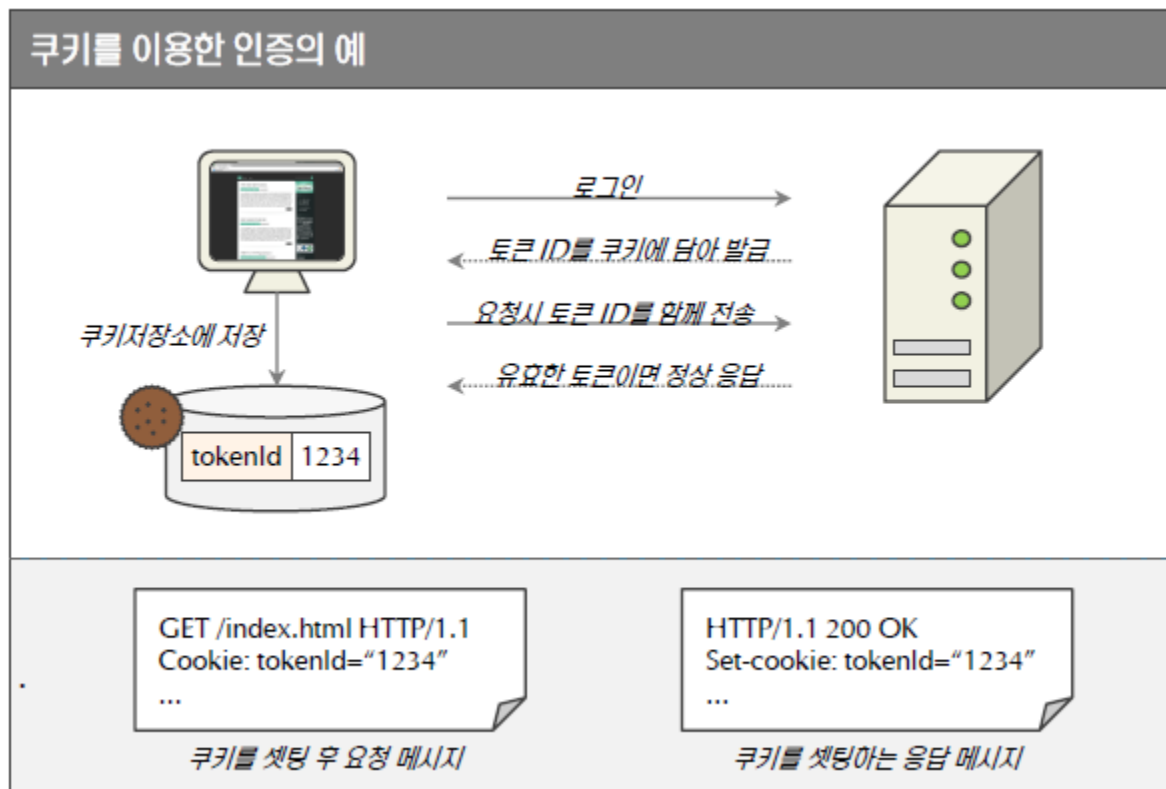
# 5.2 캐싱 [2/2] – 자원의 신선도

- ✓ 캐싱을 사용할 때는 자원의 신선도를 유지하는 것이 중요합니다.
- ✓ 캐싱은 자원의 사본을 저장하여 접근하기 때문에, 원본이 갱신 되었다면 캐시에 저장된 기존 사본을 출력해서는 안됩니다.
- ✓ 개발자가 직접 특정 정보를 캐싱하여 사용할 경우 원 정보의 업데이트 및 만료 여부를 체크해서 신선도를 유지합니다.
- ✓ Cache-Control과 Expires 등의 헤더 속성을 활용하여 캐시 사용 제한 및 사본의 만료를 알릴 수 있습니다.



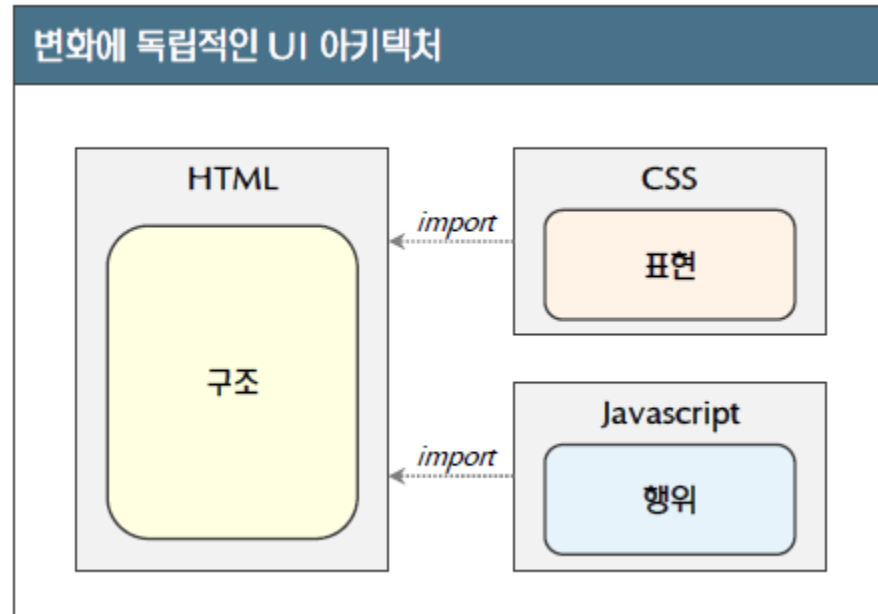
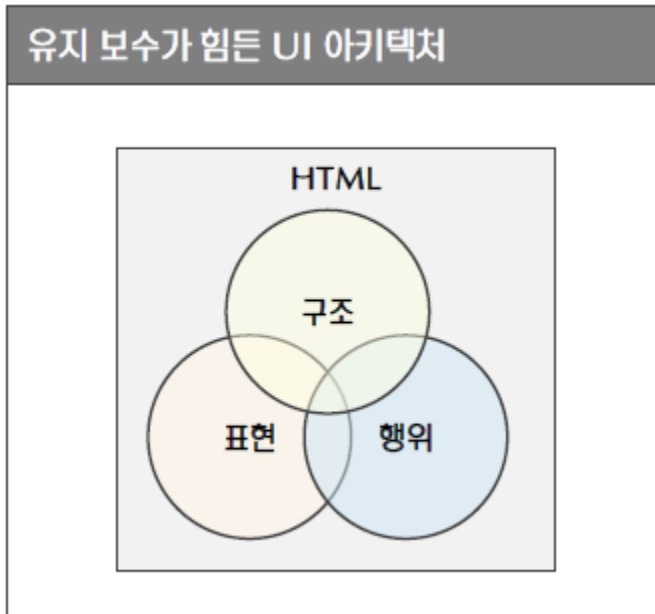
## 5.3 쿠키

- ✓ 상태를 유지하지 못하는 HTTP의 특성을 보완하기 위해, 쿠키를 사용해서 사용자를 식별하고 세션을 유지합니다.
- ✓ 쿠키는 서버에서 클라이언트로 보내는 라벨과 같으며, 웹 브라우저는 각자의 방식대로 쿠키를 저장하고 관리합니다.
- ✓ 클라이언트는 쿠키를 이용해 사용자에게 로그인 상태 유지 및 아이디 기억 등의 편의성을 제공합니다.
- ✓ 서버는 클라이언트에게 보낸 쿠키를 이용해 클라이언트를 식별하고 자원에 대한 접근을 허가할 수 있습니다.



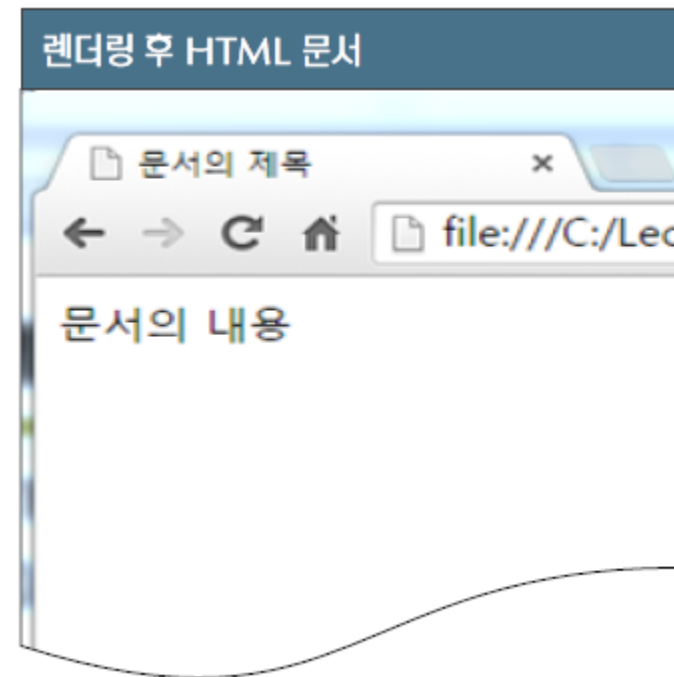
## 5.4 브라우저 페이지 (1/4) – UI 아키텍처

- ✓ 오늘날 수준 높은 요구 사항, 다양한 디바이스 등으로 UI(User Interface) 기술 구조는 점점 복잡해 지고 있습니다.
- ✓ 복잡하고 방대해진 UI 코드는 관리를 어렵게 합니다. 크로스 브라우징 및 멀티 디바이스는 코드 파편화를 가져 옵니다.
- ✓ 이러한 문제는 웹 페이지의 구조(HTML)와 행위(JS) 그리고 표현(CSS)를 분리하면 어느 정도 해결할 수 있습니다.
- ✓ 또한, 검증된 프레임워크와 라이브러리를 사용하여 시스템의 안정성과 유지 보수성을 높일 수 있습니다.



## 5.4 브라우저 페이지 (2/4) – HTML

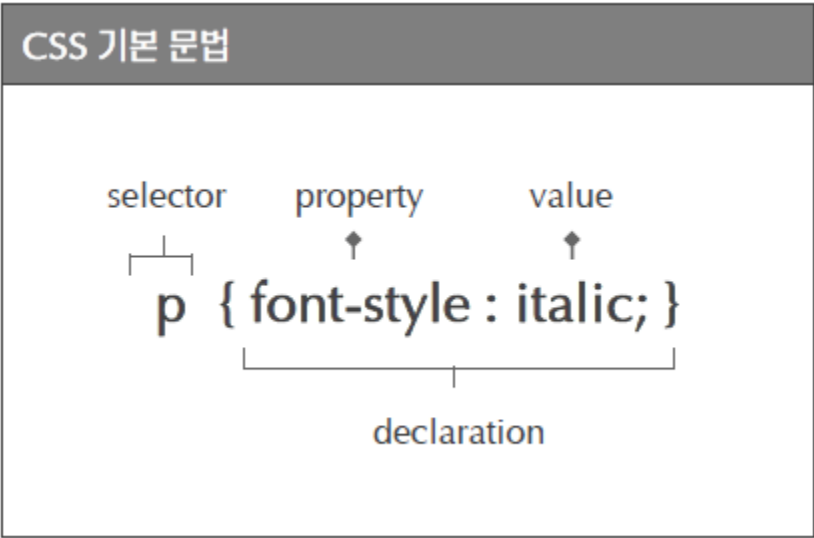
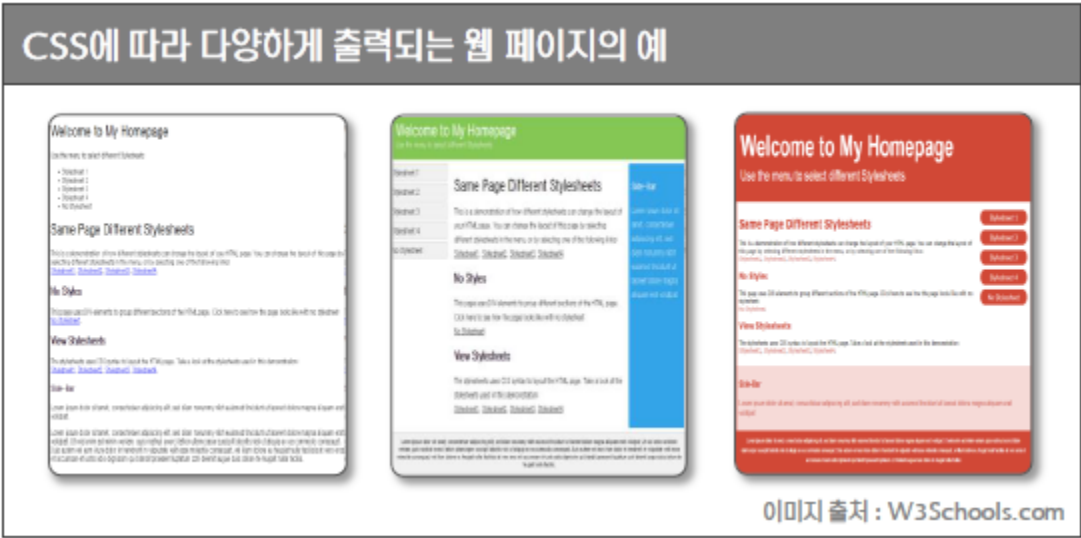
- ✓ HTML(Hypertext Markup Language)은 웹 페이지의 구조를 담당하는 언어입니다.
- ✓ SGML(Standard Generalized Markup Language)에서 파생된 언어이며 웹에 최적화된 구성을 갖추었습니다.
- ✓ 여는 태그(Tag)와(< >) 닫는 태그(</>)로 구성되며, 태그 사이에 문서의 내용을 작성합니다.
- ✓ 각 태그는 고유의 의미를 가지고 있으며, 웹 브라우저는 이 의미에 따라 문서를 화면에 출력합니다.





# 5.4 브라우저 페이지 (3/4) – CSS

- ✓ CSS(Cascading Style Sheets)란 구조적인 마크업 문서가 화면에 출력되는 모습을 정의하는 언어입니다.
- ✓ 웹을 사용하는 클라이언트들이 다양하게 발전하면서 디바이스마다 최적화된 페이지를 제공할 필요가 생겼습니다.
- ✓ HTML이 스타일까지 구현할 경우 유지 보수성이 떨어지므로, W3C에서 스타일을 담당할 언어인 CSS를 만들었습니다.
- ✓ CSS를 통해 어떤 스타일을 적용하느냐에 따라 하나의 구조를 전혀 다른 페이지처럼 꾸밀 수 있습니다.



## 5.4 브라우저 페이지 (4/4) – 자바스크립트

- ✓ JavaScript는 웹 브라우저가 실행하는 함수형 프로그래밍 언어입니다.
- ✓ UI에서 사용자의 행위를 구현하는 역할을 가집니다. 기능을 수행하지만 웹 서버가 아닌 클라이언트에서 실행됩니다.
- ✓ <script> 태그를 이용하여 HTML 내부에 작성하는 방법과 외부에 작성한 후 src 속성으로 연결하는 방법이 있습니다.
- ✓ 자바스크립트를 이용하여 웹 페이지의 DOM(Document Object Model)을 다양하게 제어할 수 있습니다.

### JavaScript 코드를 HTML 문서에 포함하는 방법

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JavaScript 선언</title>
<script type="text/javascript">
    // Some JavaScript Code here!
    function hello(target) {
        alert('Hello,' + target);
    }

    hello('JavaScript');
</script>
</head>
<body>
</body>
</html>
```

### 외부 JavaScript 파일을 HTML 문서에 포함하는 방법

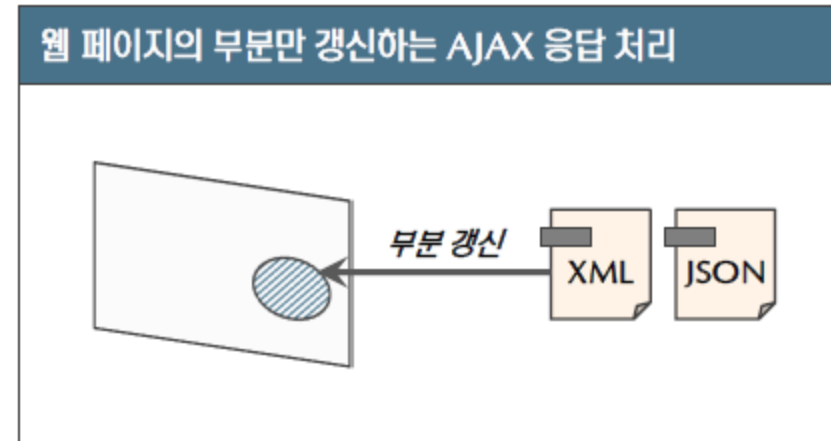
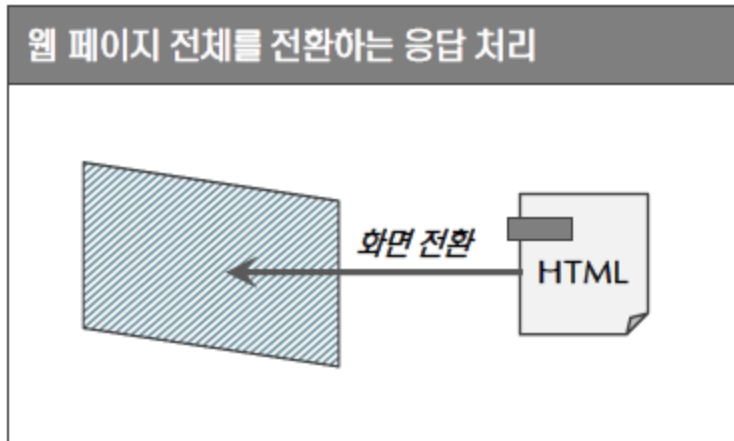
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JavaScript 선언</title>
<script src="hello.js" type="text/javascript">
</script>
</head>
<body>
</body>
</html>
```

```
function hello(target) {
    alert('Hello,' + target);
}

hello('JavaScript');
```

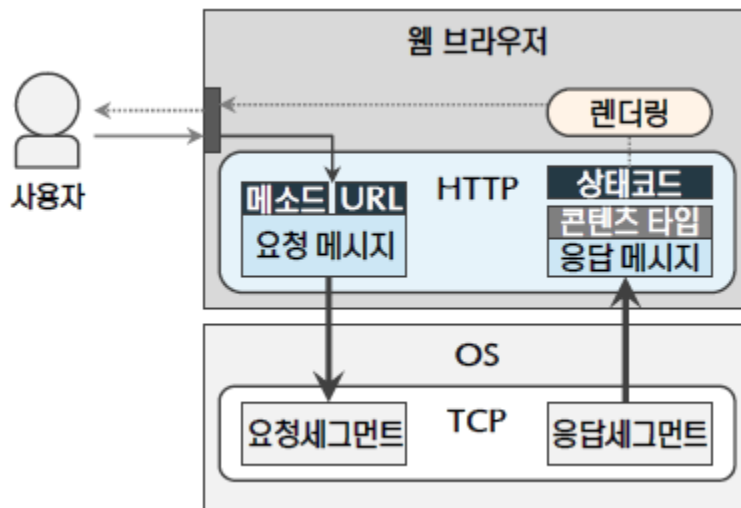
## 5.5 AJAX

- ✓ AJAX(Asynchronous Javascript and XML)는 요청에 대한 응답으로 자원의 일부분만 갱신하는 처리 방법입니다.
- ✓ 일반적으로는 사용자의 요청에 대한 응답에 따라 화면이 이동되거나 새로 초기화되면서 처리 결과가 반영됩니다.
- ✓ 하지만 AJAX는 자바스크립트를 이용해 전달받은 처리 결과를 DOM의 일부분만 갱신합니다.
- ✓ AJAX는 주로 XML, JSON 그리고 순수 텍스트를 요청-응답의 콘텐츠로 사용합니다.



## 5.6 요약

- ✓ 웹 브라우저는 HTTP 통신의 클라이언트 역할을 수행합니다. 즉, 요청을 보내고 응답을 받아 사용자에게 제공합니다.
- ✓ 웹 브라우저는 서버로부터 받은 응답 메시지를 콘텐츠 타입에 맞게 렌더링하여 화면에 출력합니다.
- ✓ 브라우저 페이지를 구성하는 언어는 HTML, CSS, 자바스크립트가 있으며 각각 역할에 맞게 분리하여 작성합니다.
- ✓ AJAX를 이용하면 화면 전환 없이 응답 결과를 반영할 수 있습니다.
- ✓ 또한 자원을 캐싱하여 응답 속도를 높일 수 있으며, 서버가 보낸 쿠키 정보를 이용하여 인증, 상태 유지 등에 활용합니다.





## 6. 웹 애플리케이션 아키텍처

---

- 6.1 웹 애플리케이션 아키텍처 개요
- 6.2 웹 애플리케이션 개발 방법
- 6.3 웹 서비스
- 6.4 요약

# 6.1 웹 애플리케이션 아키텍처 개요

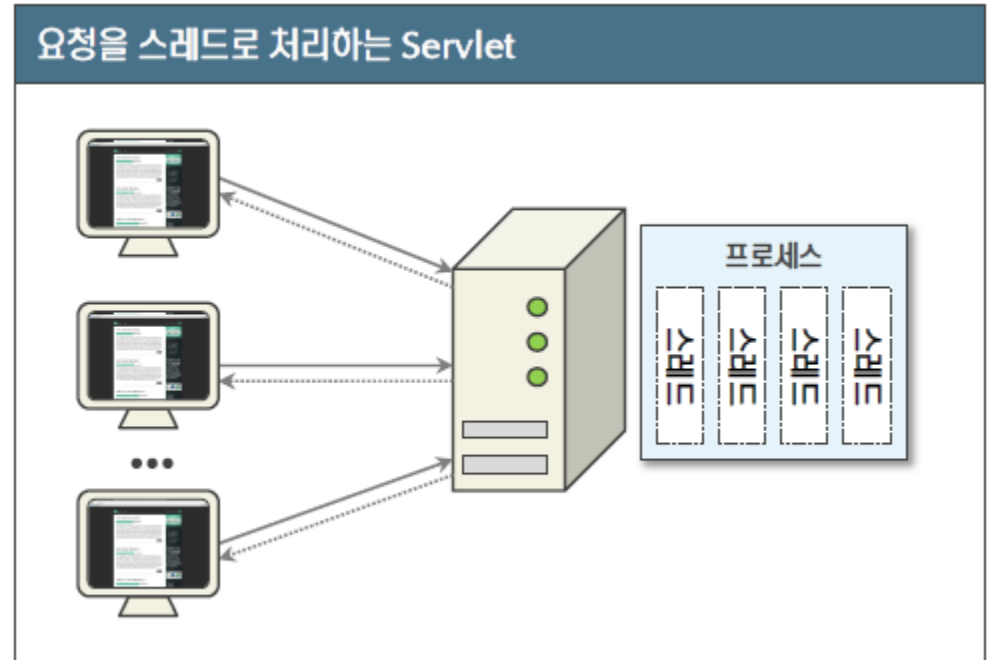
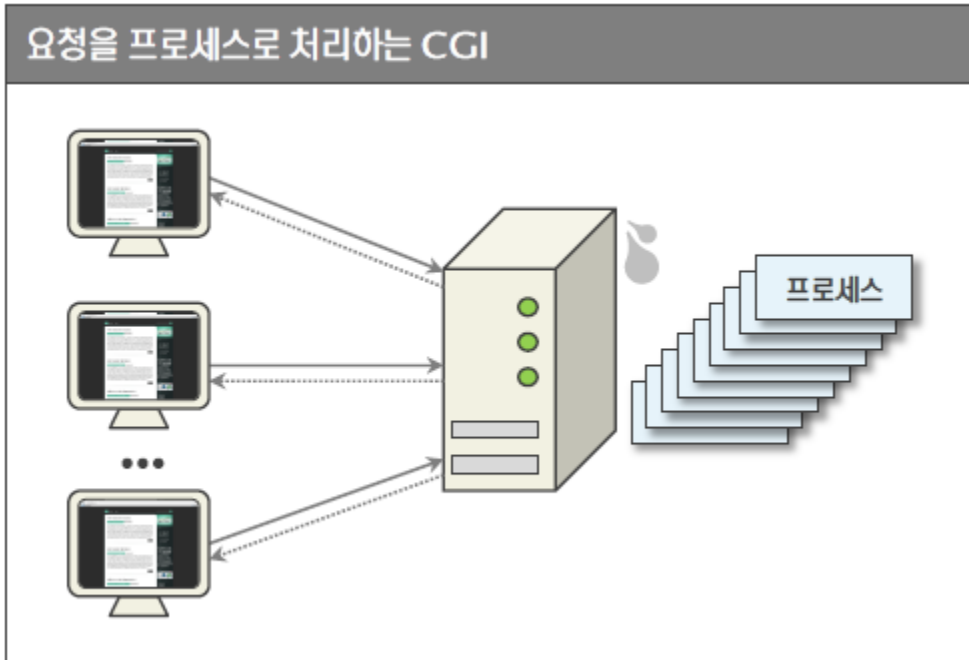
- ✓ 사용자들은 점점 더 똑똑하고 편리한 웹 애플리케이션을 경험하고자 합니다. 그에 따라 기능의 복잡도가 증가하고 있습니다.
- ✓ 복잡한 기능을 수행하는 로직의 몸집은 점점 커졌고, 이로 인해 작은 변경 사항이 발생해도 많은 코드를 수정해야 했습니다.
- ✓ 관심사의 분리(Separation of Concern)를 통해 코드의 중복이 적고 변경에 유연한 애플리케이션을 만들 수 있습니다.
- ✓ 레이어 설계를 통한 확장성과 이식성을 높이기 위해서 비즈니스 로직 레이어를 지키는 것이 중요합니다.





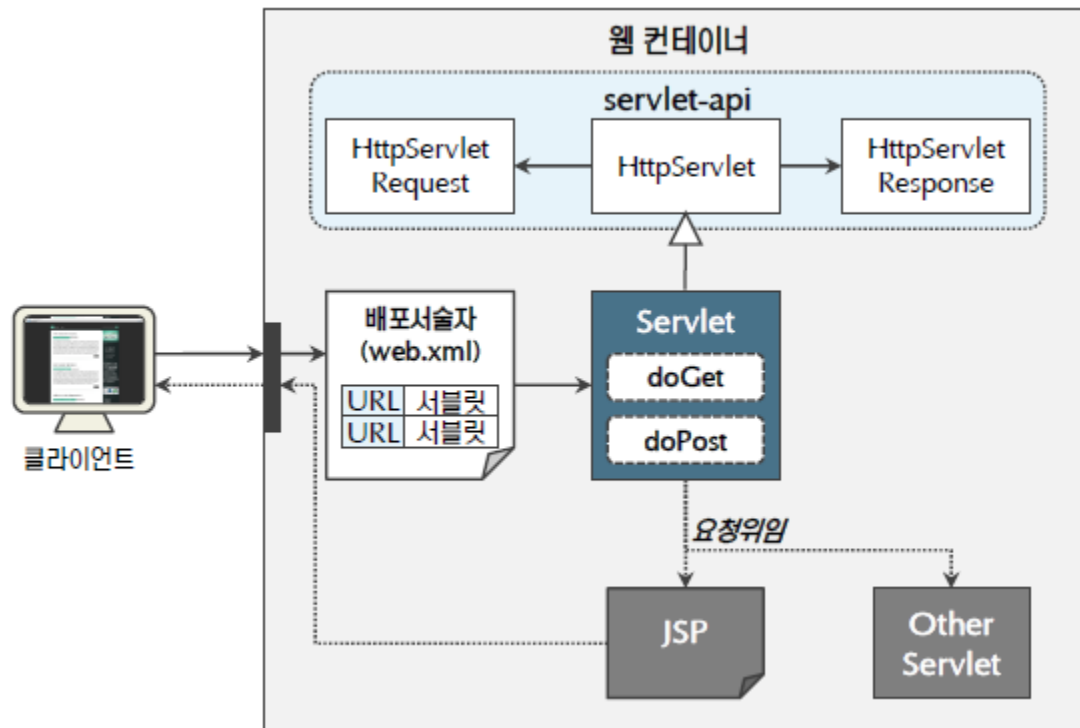
## 6.2 웹 애플리케이션 개발 방법 (1/6) – CGI

- ✓ CGI(Common Gateway Interfaces)는 서버에서 동적인 콘텐츠를 생성하여 클라이언트에 제공하는 인터페이스입니다.
- ✓ 전통적인 CGI 방식은 클라이언트 요청마다 개별 프로세스가 생성되어 시스템 부하가 많이 생기는 단점이 있었습니다.
- ✓ 또한 플랫폼에 종속적이어서 윈도우에서 C언어 등으로 만들어진 CGI 애플리케이션은 리눅스에서 사용할 수 없었습니다.
- ✓ 이러한 단점을 해결하기 위하여 개별 스레드가 요청을 처리하는 방식으로 발전하였습니다. (ASP, PHP, Servlet 등)



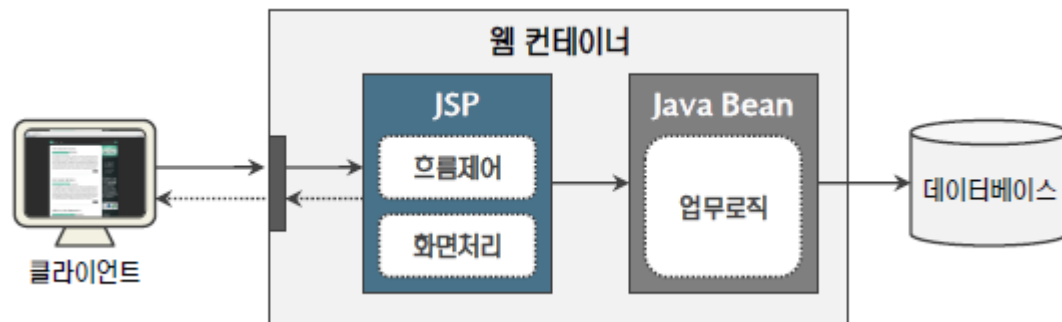
## 6.2 웹 애플리케이션 개발 방법 (2/6) – Servlet

- ✓ 요청 URL과 매칭된 서블릿을 배포 서술자(Deployment Descriptor, web.xml)를 통해 연결됩니다.
- ✓ 서블릿 스펙 3.0부터는 어노테이션을 이용해 서블릿(@WebServlet)과 필터(@WebFilter)등 연결이 가능합니다.
- ✓ 서블릿 클래스는 servlet-api의 HttpServlet을 상속받아 구현하고, 요청 메소드에 따라 기능을 구현합니다.
- ✓ 매핑된 서블릿이 요청을 직접 처리하거나 다른 서블릿 또는 JSP로 요청을 위임할 수도 있습니다.



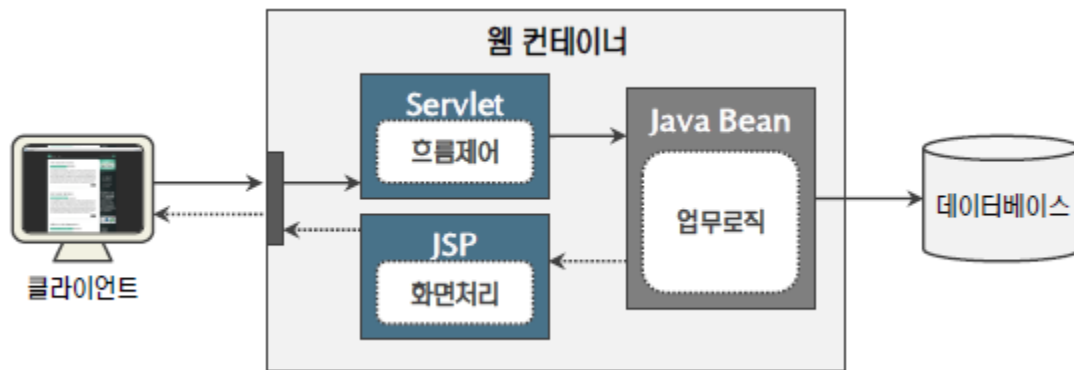
## 6.2 웹 애플리케이션 개발 방법 (3/6) – JSP

- ✓ JSP(Java Server Pages)는 HTML 문서 형식에 Java 로직을 작성할 수 있도록 제공하는 API입니다.
- ✓ 스크립트릿(<%...%>) 내에 Java 로직을 작성하여 서블릿과 같은 역할을 할 수 있습니다.
- ✓ 하지만 흐름 제어 로직과 화면 처리가 모두 JSP안에 혼재되어 있으면 요구사항이 복잡해 질수록 유지 보수가 어려워 집니다.
- ✓ HTML 코드와 Java 코드가 섞여 있으므로 디자이너와 개발자간에 원활한 협업이 어렵습니다.



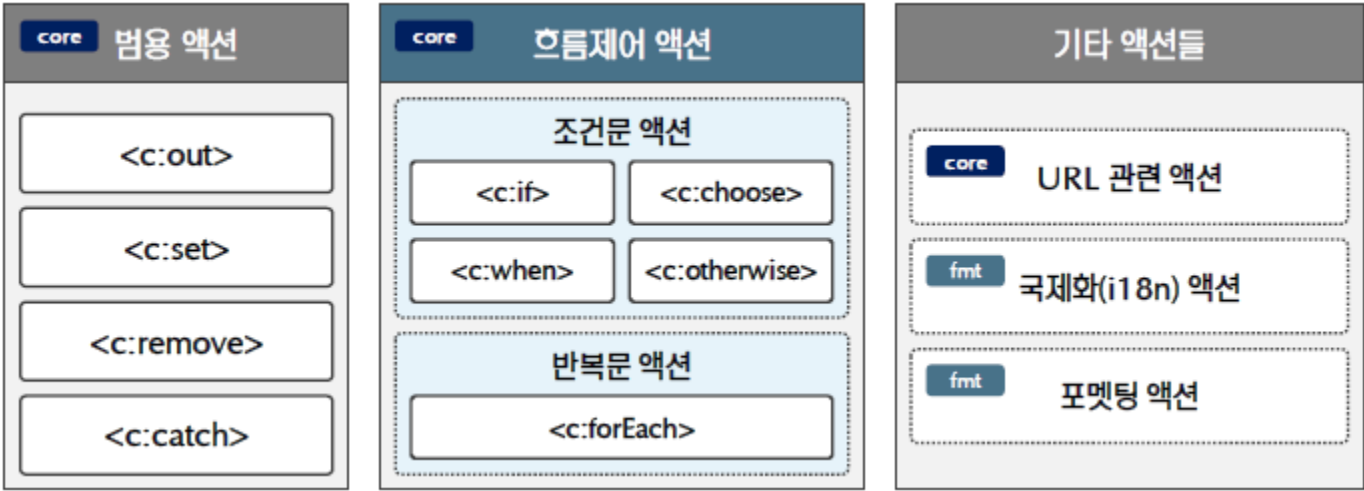
## 6.2 웹 애플리케이션 개발 방법 (4/6) – JSP Model 2

- ✓ 흐름 제어와 화면 처리를 모두 JSP에서 구현하는 모델 1 방식과 달리 모델 2는 역할에 따라 구현 공간을 분리합니다.
- ✓ 모델 2는 서블릿이 흐름 제어를 담당하고 JSP는 화면 처리를 담당하므로 디자이너와 개발자의 작업을 분리할 수 있습니다.
- ✓ 업무 로직은 Model, 화면 처리는 View, 흐름 제어는 Controller로 분배하였다고 해서 MVC 모델이라고도 부릅니다.
- ✓ 그러나 개발 초기에 아키텍처 설계를 위한 시간이 소요되어 개발 기간이 늘어날 수 있습니다.



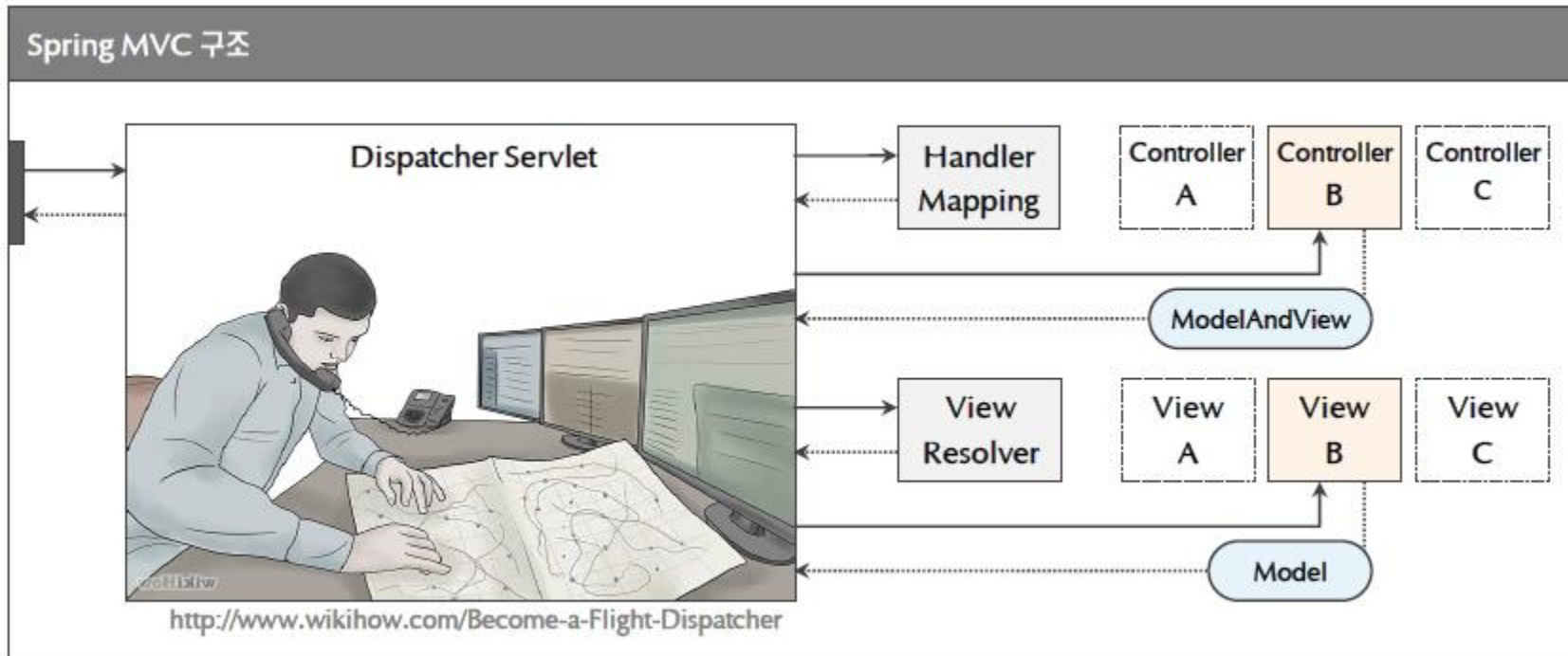
## 6.2 웹 애플리케이션 개발 방법 (5/6) – JSTL

- ✓ JSP를 사용하여 프리젠테이션 영역을 개발하는 많은 개발자는 Java와 같은 프로그래밍 언어에 익숙하지 않습니다.
- ✓ 그러나, JSP 페이지에서 동적인 데이터를 처리하기 위해서는 스크립트릿에 프로그래밍 언어를 사용해야 합니다.
- ✓ JSTL(Java Standard Tag Library)은 JSP에서 스크립트릿에 Java 코딩을 하지 않아도 로직을 구현할 수 있습니다.
- ✓ JSTL에는 다양한 액션을 위한 태그들이 정의되어 있어, EL과 함께 사용하여 코드를 간결하게 작성할 수 있습니다.



## 6.2 웹 애플리케이션 개발 방법 (6/6) – Spring MVC

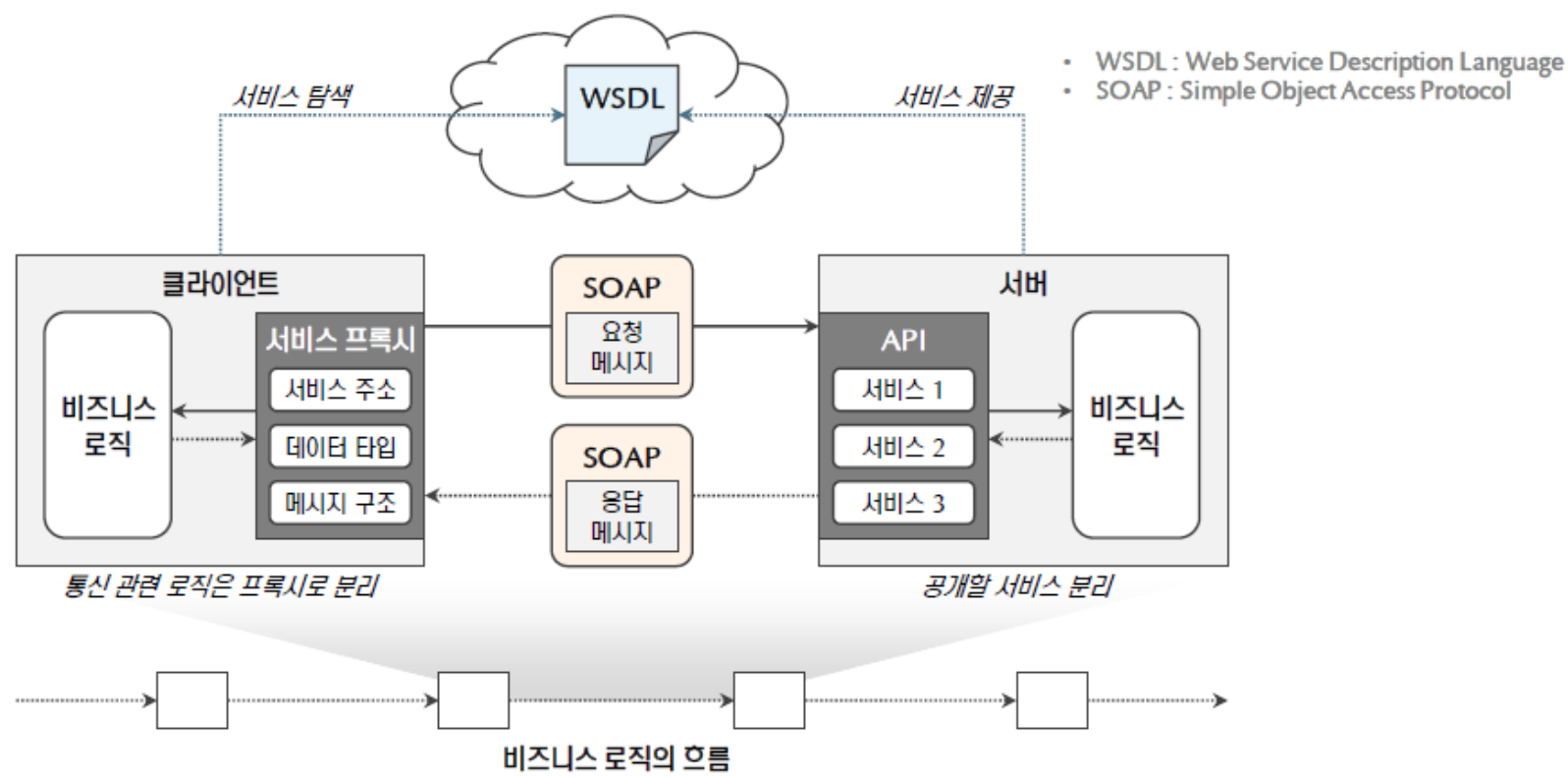
- ✓ Spring MVC는 MVC 패턴 기반의 웹 프레임워크입니다. 즉, 역할에 따라 개발 영역이 분리되어 있습니다.
- ✓ Dispatcher Servlet이 모든 요청을 받아 각 컨트롤러로 요청을 위임합니다. 이러한 패턴이 프론트 컨트롤러 패턴입니다.
- ✓ 설정보다는 관례(CoC) 중심 프레임워크로서 보편적인 기능은 기본 제공하고 그 외 기능을 설정을 통해 확장 가능합니다.
- ✓ Spring @MVC라고 불리울 만큼 어노테이션을 이용한 편리하고 효율적인 개발을 지원합니다.





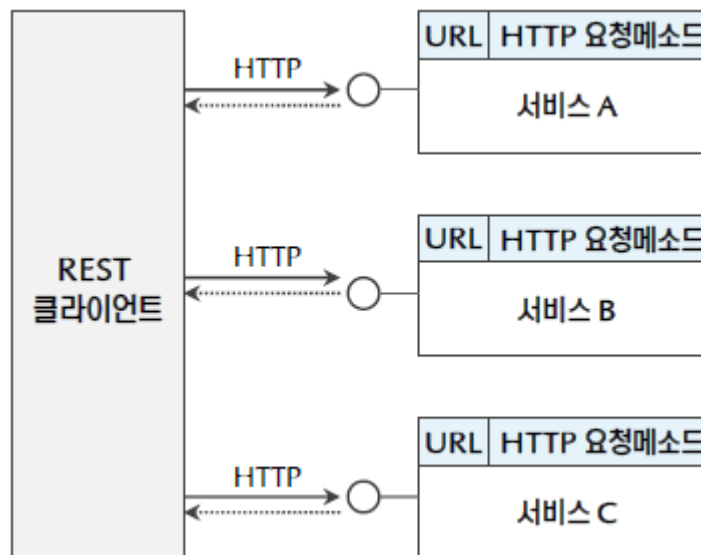
## 6.3 웹 서비스 (1/2) – WSDL/SOAP

- ✓ 웹 서비스는 비즈니스 로직의 흐름에서 웹 통신을 이용해 외부 시스템의 서비스에 접근하는 것입니다.
- ✓ 웹 서비스를 적용하면 클라이언트와 서버의 결합도를 낮추어 독립적인 진화를 이룰 수 있습니다.
- ✓ 이러한 아키텍처를 서비스 중심 아키텍처(SOA, Service Oriented Architecture)라고 합니다.
- ✓ 서버는 공개할 서비스들을 WSDL을 통해 공개하고, 클라이언트는 WSDL에서 서비스를 탐색하여 접근합니다.
- ✓ 클라이언트와 서버는 XML 기반인 SOAP 메시지 구조를 이용해 요청-응답 메시지를 주고 받습니다.



## 6.3 웹 서비스 (2/2) – REST

- ✓ REST(REpresentational State Transfer)는 웹에 서비스를 발행하고 접근하는 웹 서비스 방식 중 하나입니다.
- ✓ 별도의 전송 계층 없이 HTTP 요청 메시지 형식만으로 발행된 서비스에 연결하여 구조와 방법이 간결합니다.
- ✓ RESTful API는 마치 브라우저에서 URL만으로 자원을 조회하듯이 비즈니스 서비스에 접근합니다.
- ✓ RESTful API는 JSON, XML, 일반텍스트 등을 요청-응답 형식으로 사용합니다.



# 6.4 요약

- ✓ 관심사의 분리(Separation Of Concern)를 통해 코드의 중복이 적고 변경에 유연한 애플리케이션을 만들 수 있습니다.
- ✓ 웹 애플리케이션을 개발 방법은 점차 진화하고 있고, 프레임워크를 사용하면 쉽고 빠르게 개발할 수 있습니다.
- ✓ 요구사항의 복잡도가 증가하면서 요청 연결과 화면 처리 그리고 비즈니스 로직을 분리하는 것이 효율적입니다.
- ✓ 비즈니스의 흐름을 구성할 때 웹 서비스를 이용하면 좀 더 독립적인 시스템을 설계할 수 있습니다.

UI 레이어	프리젠테이션 레이어	서비스 발행 레이어	비즈니스 로직 레이어	데이터 접근 레이어	데이터 영역
<div>JavaScript</div> <div>HTML</div> <div>CSS</div> <div>jQuery</div> <div>Bootstrap</div> <div>Angular JS</div>	<div>JSP/Servlet</div> <div>Struts</div> <div>SpringMVC</div>	<div>Jersey</div> <div>AXIS2</div> <div>Apache CXF</div>	<div>POJO</div> <div>Spring</div> <div>EJB</div>	<div>JDBC</div> <div>MyBatis</div> <div>JPA</div>	<div>RDB</div> <div>NoSQL</div> <div>File</div>
Front-end 개발자(UI개발자)의 활동영역			Back-end 개발자(서버개발자)의 활동영역		

# 토론

- ✓ 질문과 대답
- ✓ 토론

