과목명: 웹프로그래밍응용

교재: React.js, 스프링 부트, AWS로 배우는 웹개발 101(2판)

Ch01. 개발을 시작하기 전에

Professor Seung-Hoon Choi

덕성여자대학교
DUKSUNG WOMEN'S UNIVERSITY

목차

1.1 이 책을 읽는 방법

1.1.1 예제와 실습 코드

- 예제
  - 읽거나 확인만 하면 됨
- 실습 코드
  - 직접 작성할 것

1.1 이 책을 읽는 방법

1.1.2 소스 코드

- 깃허브
  - https://github.com/fsoftwareengineer/todo-application-revision2.git

1.1.3 커맨드라인 인터페이스

- Command Line Interface
  - 도구나 운영체제에 대한 의존성이 낮음
  - 개발자라면 익숙해져야 함

1.2 Todo 웹 애플리케이션

1.2.1 Todo 웹 애플리케이션 기능

그림 1-1 Todo 웹 애플리케이션

1.2 Todo 웹 애플리케이션

1.2.1 Todo 웹 애플리케이션 기능

- Todo 생성 : + 버튼을 눌러 Todo 아이템을 생성할 수 있다.
- Todo 리스트 : 생성된 아이템 목록을 화면에서 확인할 수 있다.
- Todo 수정 : Todo 아이템을 체크하거나 내용을 수정할 수 있다.
- Todo 삭제 : Todo 아이템을 삭제할 수 있다.
- 회원가입 : 사용자는 애플리케이션에 회원가입하고, 생성된 계정을 이용해 Todo 애플리케이션에 접근할 수 있다.
- 로그인 : 계정을 생성한 사용자는 계정으로 로그인할 수 있다.
- 로그아웃 : 로그인한 사용자는 로그아웃할 수 있다.

1.2.2 Todo 웹 애플리케이션 아키텍처

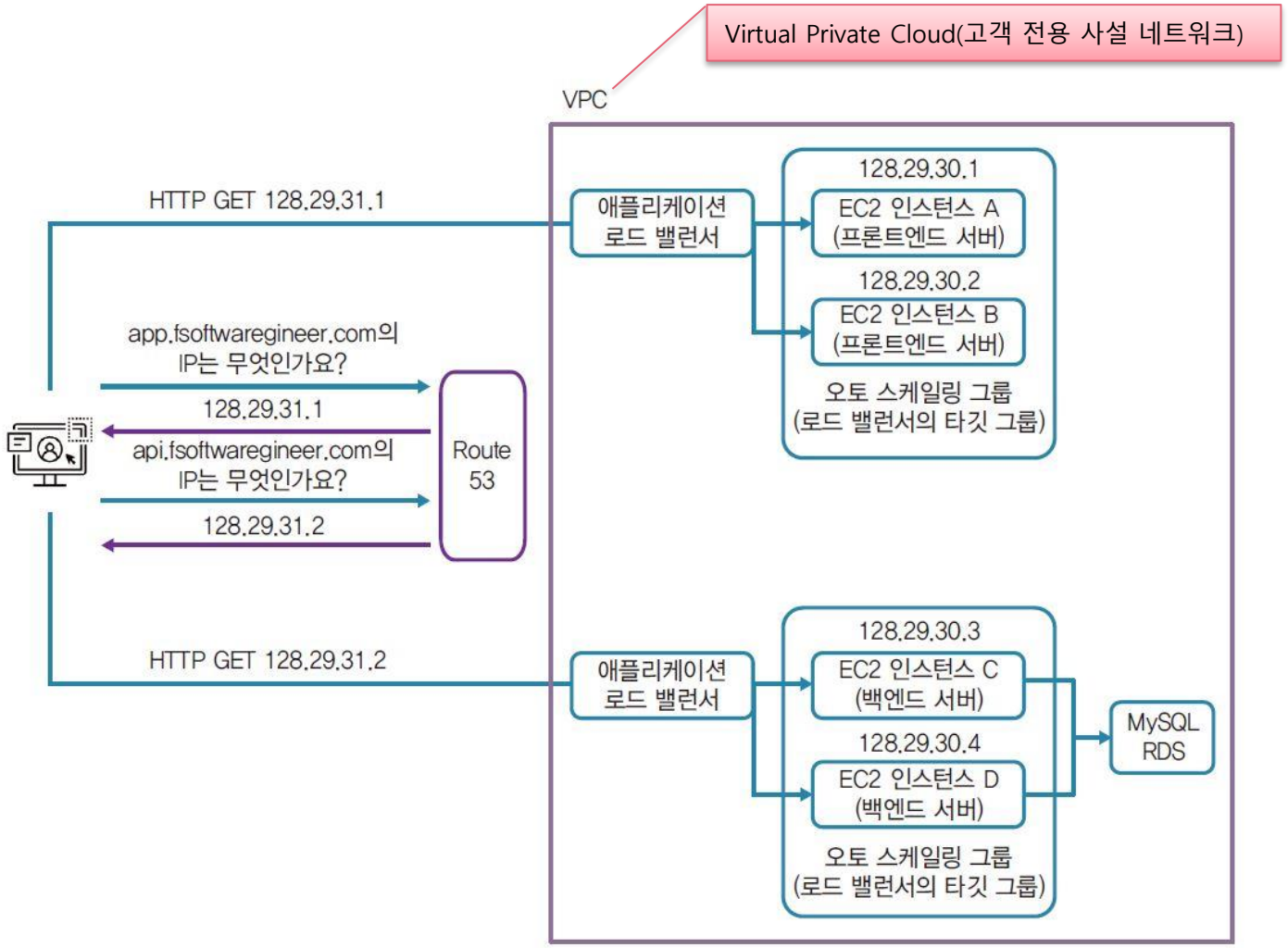

그림 1-2 배포할 애플리케이션의 아키텍처

1.2.3 기술과 구현 사이

- HTML/CSS/React.js : 프론트엔드 애플리케이션 개발에 사용한다. 우리의 프론트엔드 애플리케이션은 프론트엔드 클라이언트를 반환하는 서버가 있다. 이 서버의 역할은 단 하나이다. 바로 React.js 애플리케이션을 반환하는 것이다. 이런 방식으로 프론트엔드와 백엔드를 분리decouple할 수 있다. 왜 분리할까? 생각해보자.

1.2.3 기술과 구현 사이

- 스프링 부트 : 백엔드 애플리케이션 개발에 사용한다. 우리는 스프링 부트로 REST API를 구현한다. 이 API는 프론트엔드 애플리케이션이 사용한다. 프론트엔드 애플리케이션은 꼭 웹 애플리케이션일 필요는 없다. 예를 들어 웹 애플리케이션 배포 이후 모바일 앱을 만든다고 하자. 모바일 앱 역시 별도의 백엔드 개발 없이 백엔드 애플리케이션의 REST API를 사용할 수 있다. REST API를 구현하고 프론트엔드를 분리하는 것은 이후 마이크로서비스 아키텍처로 서비스를 확장하는 데 용이하다.

1.2.3 기술과 구현 사이

- AWS : 프론트엔드와 백엔드 애플리케이션이 실행될 프로덕션 환경을 구축하기 위해 사용한다. 다행히도 우리는 AWS 내에서 필요한 모든 것을 구축할 수 있다. 구체적으로 어떤 기술을 사용할지는 6장에서 설명하도록 한다.

1.3 배경 지식

1.3.1 하이퍼텍스트 트랜스퍼 프로토콜(HTTP)

- HyperText Transfer Protocol
  - 하이퍼텍스트를 주고받기 위한 통신 규약

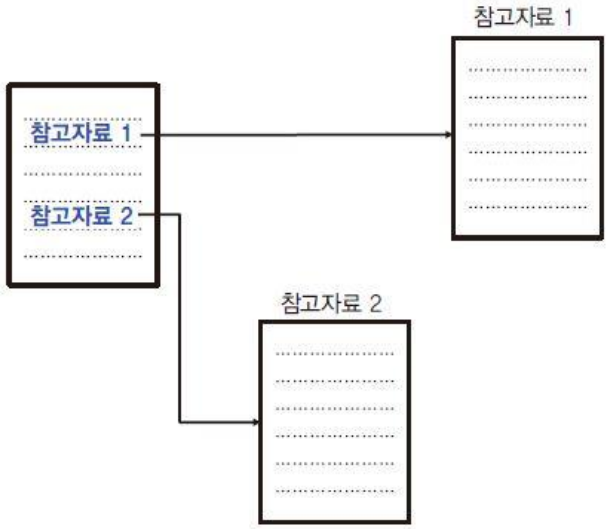

그림 1-3 하이퍼텍스트

1.3.1 하이퍼텍스트 트랜스퍼 프로토콜(HTTP)

HTTP 응답을 렌더링
(화면에 디스플레이)함

그림 1-4 HTTP 요청과 응답

요청 메소드

예제 1-1. HTTP 요청

```
GET / HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Upgrade-Insecure-Requests: 1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/605.1.15 (KHTML, like
Gecko) Version/14.0.3 Safari/605.1.15
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

1.3.1 하이퍼텍스트 트랜스퍼 프로토콜(HTTP)

요청 시 메소드 종류

- 요청 시 호스트에게 지정한 리소스에 대해 어떤 작업을 원하는지 알려줌

표 1-1 HTTP 메서드와 기능

메서드	기능
GET	리소스를 가져올 때 사용
POST	리소스에 대해 임의의 작업(예, 생성, 수정)을 할 때 사용
PUT	리소스를 대체할 때 사용
DELETE	리소스를 삭제할 때 사용

일부분 수정 시 PATCH 메소드를 이용하기도 함

1.3.1 하이퍼텍스트 트랜스퍼 프로토콜(HTTP)

[예제 1-2. HTTP 응답]

```
HTTP/1.1 200
Content-Type: text/html;charset=UTF-8
Keep-Alive: timeout=60
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Date: Sat, 17 Apr 2021 05:28:42 GMT
Content-Length: 32
Connection: keep-alive
X-Frame-Options: DENY
Vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers

<html></html>
```

응답 코드

컨텐트 타입

응답 바디

1.3.1 하이퍼텍스트 트랜스퍼 프로토콜(HTTP)

- **응답 코드**
  - 200: 성공적으로 요청을 처리함
  - 404: 해당 리소스가 존재하지 않음
  - 403: 해당 리소스에 접근할 권한이 없음
  - 500: 서버의 오류로 요청을 처리할 수 없음

- **Content-type**
  - 응답의 미디어 타입
  - 예: text/html, text/css, application/json, video/mpeg 등

- **응답 바디**
  - 요청 결과를 담는 곳

1.3.2 자바스크립트 오브젝트 노테이션(JSON)

- **JSON**
  - JavaScript Object Notation
  - '오브젝트'를 표현하기 위한 문자열

- **오브젝트**
  - 메모리 상에 존재하는 자료 구조

[예제 1-3. 자바 TodoItem 클래스]

```
public class TodoItem {
 String title;
 boolean done;

 public TodoItem(String title, boolean done) {
  this.title = title;
  this.done = done;
 }
}
```

1.3.2 자바스크립트 오브젝트 노테이션(JSON)

[예제 1-4. TodoItem 오브젝트]

```
new TodoItem("myTitle", false);
```

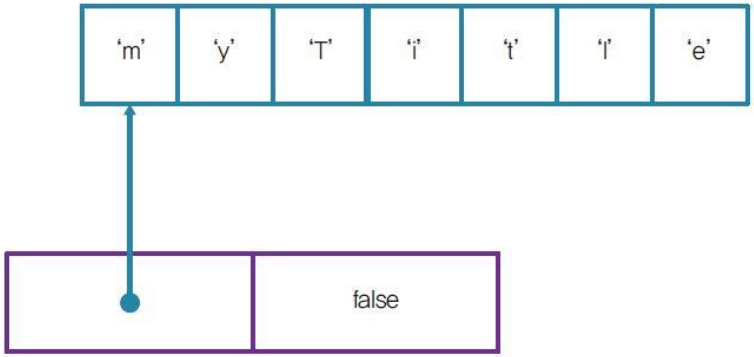

그림 1-5 메모리상의 오브젝트

1.3.2 자바스크립트 오브젝트 노테이션(JSON)

- **인터넷을 통해 TodoItem 오브젝트 전송 시**
  - 메모리상의 오브젝트를 애플리케이션1과 애플리케이션2가 모두 이해할 수 있는 형태로 변환해야 함

- **직렬화(serialization)**
  - 메모리상의 오브젝트를 전송 시의 형태로 변환하는 작업

- **역직렬화(deserialization)**
  - 직렬화의 반대 작업

그림 1-6 인터넷을 이용해 데이터를 교환하는 애플리케이션

1.3.2 자바스크립트 오브젝트 노테이션(JSON)

JSON

- 키-값(key-value)의 형태로 오브젝트를 표현한 문자열

[예제 1-5. JSON 형태의 TodoItem 오브젝트]

```
{
   "title":"myTitle",
   "done":false
}
```

표 1-2 JSON에서 자료형 표현 방법

자료형/구조	표현 방법
Boolean	true 또는 false
숫자	쌍따옴표 없는 숫자. 예) 10, 52.2 등
문자열	쌍따옴표로 감싼 형태. 예) “abc”, “myTitle” 등
오브젝트	중괄호로 감싼 형태. 예) { “title” : “myTitle” }
배열	대괄호로 감싼 형태. 예) [“abc”, “myTitle”] 등

1.3.2 자바스크립트 오브젝트 노테이션(JSON)

[예제 1-6. JSON 예]

```
{
  "myString":"hello", // 문자열
  "number":10, // 숫자
  "myStringArray":[ // 문자열 배열
    "abc",
    "def"
  ],
  "myObject":{ // 오브젝트
    "name":"obj1"
  }
}
```

1.3.2 자바스크립트 오브젝트 노테이션(JSON)

- 자바스크립트에서 오브젝트 생성하는 방법과 JSON이 동일함

[예제 1-7. 자바스크립트에서 오브젝트 생성하는 방법]

```
var object = {
 "title": "myTitle",
 "done": false
};
```

1.3.2 자바스크립트 오브젝트 노테이션(JSON)

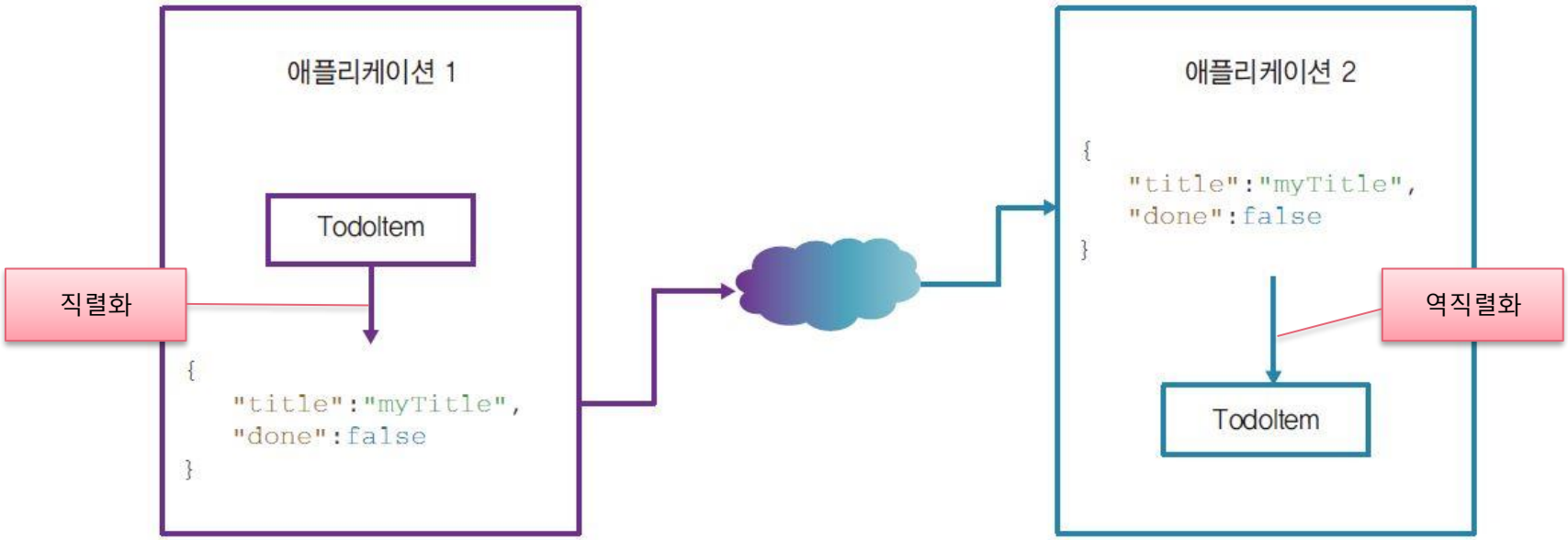

그림 1-7 JSON을 이용해 데이터 전달

1.3.2 자바스크립트 오브젝트 노테이션(JSON)

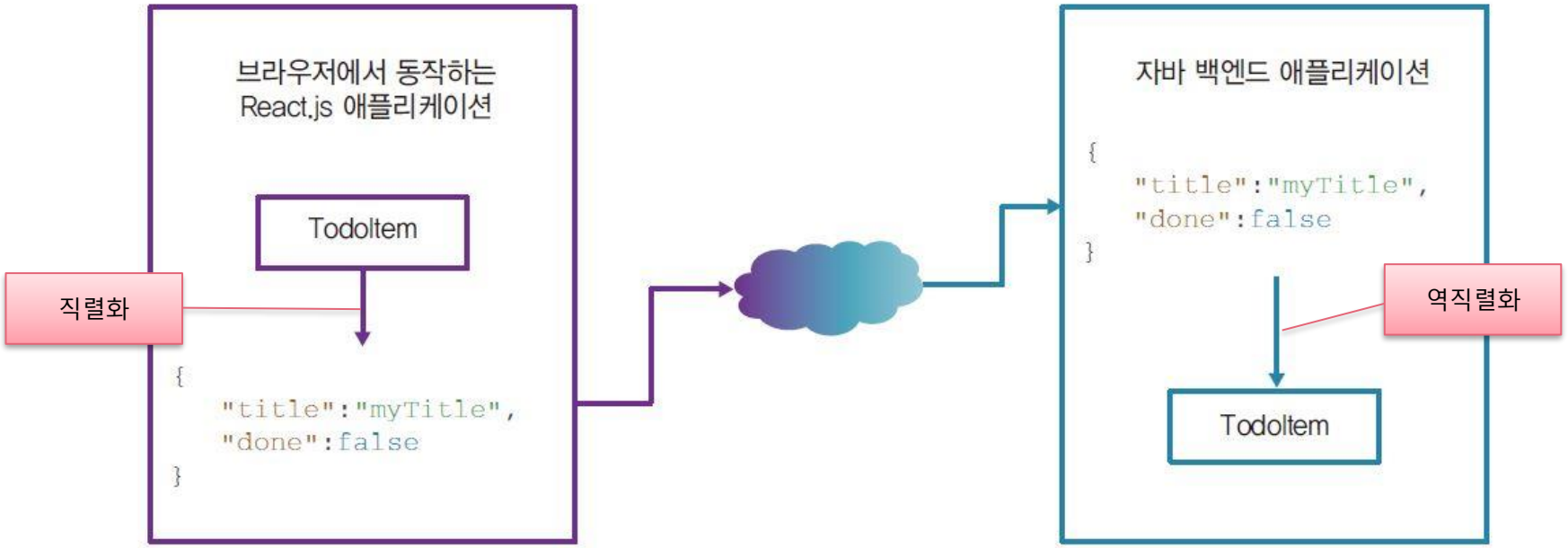

그림 1-8 Todo 프로젝트에 적용한 JSON 통신

1.3.3 서버란?

- **서버**
  - 일종의 프로그램
  - 지정된 포트(예: 8080)에 소켓을 열고 클라이언트가 연결할 때까지 무한히 대기함
  - 클라이언트가 연결하면 클라이언트 소켓에서 요청을 받아 일을 수행하고 응답을 작성해서 보내줌

- **예: FTP 서버(파일 송수신 서버)**
  - 클라이언트 서버 통신 시 File Transfer Protocol을 사용함

20번 포트

- **예: HTTP 서버(웹 서버)**
  - 클라이언트 서버 통신 시 Hyper Text Transfer Protocol을 사용함

80 또는 8080 번 포트

[예제 1-8. 아주 간단한 서버 예]

```
import java.net.ServerSocket;
import java.net.Socket;

public class WebServer {

 public static void main(String[] args) {
  new WebServer().run();
 }

 public void run() {
  try {
   ServerSocket serverSocket = new ServerSocket(8080);
   while (true) {
    try {
     Socket client = serverSocket.accept();
     new Thread(() -> handleClient(client)).start();
    } catch (Exception e) {
     e.printStackTrace();
    }
   }
  } catch (Exception e) {
   e.printStackTrace();
  }
 }

 private static void handleClient(Socket client) {
  // (1) 클라이언트의 요청 읽어오기
  // (2) 클라이언트의 요청에 맞는 작업 수행하기
  // (3) 클라이언트에게 응답 작성하기
  // (4) 소켓 닫기
 }
}
```

소켓: 네트워크 상에서 실행되는 두 개의 프로그램 간 양방향 통신을 위한 엔드포인트(창구)

포트 번호

여기서 무한 대기하다가,
클라이언트로부터 요청이 오면
클라이언트에 대한 소켓을 얻음

람다 구문을 이용하여
Runnable 인터페이스의
유일한 메소드인 run( )에 대한
바디 부분을 제공함

1.3.4 정적 웹 서버

- **정적 웹 서버(Static Web Server)**
  - 아무런 부가적인 작업없이 리소스 파일을 있는 그대로 리턴하는 웹 서버
  - 예: 아파치, Nginx 등
  - 교재에서는, 리액트를 반환하는 프론트엔드 서버가 정적 웹 서버임

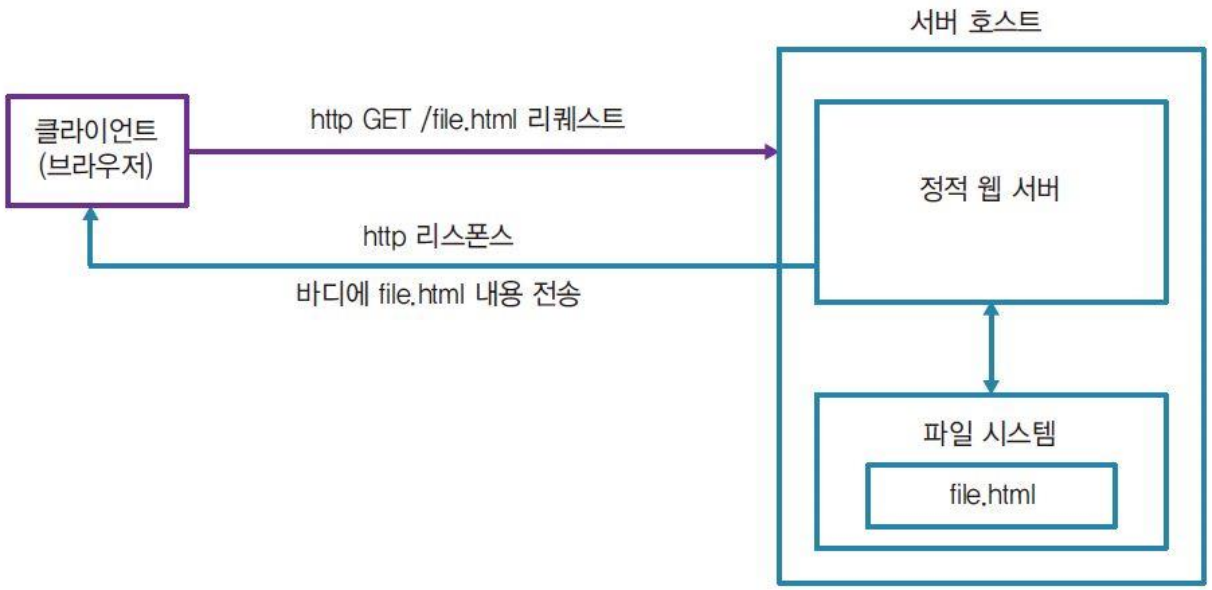

그림 1-9 정적 웹 서버가 하는 일

1.3.5 동적 웹 서버

동적 웹 서버(Dynamic Web Server)

- 요청을 처리한 후 처리한 결과에 따라 응답 바디를 재구성하거나 HTML 템플릿 파일(예: jsp 파일)에 적용한 결과를 보냄
- 클라이언트가 누군지, 어떤 매개변수를 보내는지에 따라 다른 응답을 받음
- 아파치 톰캣
  - 동적인 웹 서버 구현을 도와주는 프로그램
  - 서블릿 엔진이라고 함

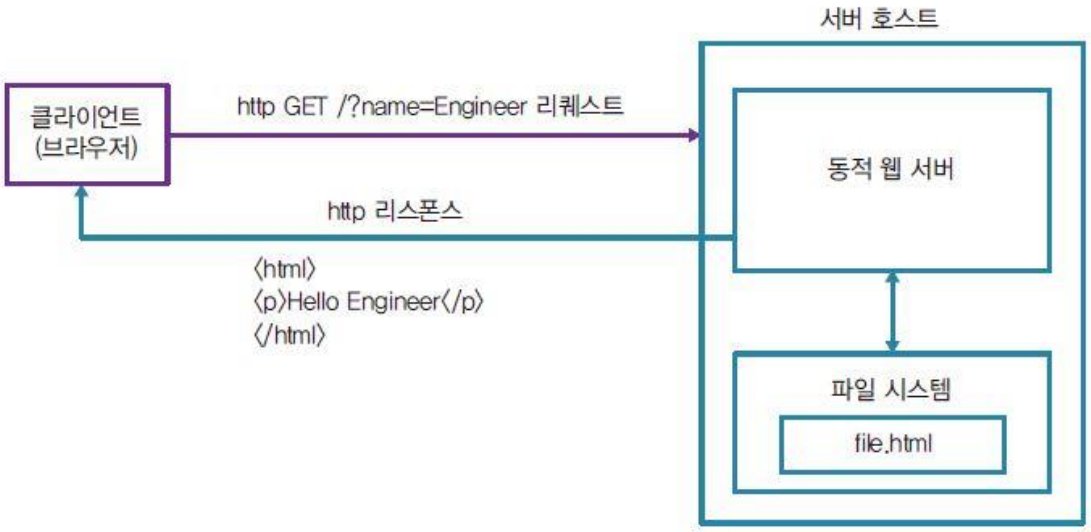

그림 1-10 동적 웹 서버의 예

1.3.6 자바 서블릿 컨테이너/엔진

- **서블릿 컨테이너/엔진**
  - 서버에서 동작하는 프로그램
  - 내부에 서블릿(특정 자바 클래스)이 실행됨
- **서블릿**
  - 서블릿 컨테이너에서 동작하는 클래스(객체)
  - javax.servlet.http.HttpServlet 클래스를 상속받아 작성됨
  - 개발자는 서블릿 안에 비즈니스 로직을 구현함
- **스프링 부트도 내부적으로 서블릿 엔진의 사용을 위해 서블릿을 상속 및 구현함**

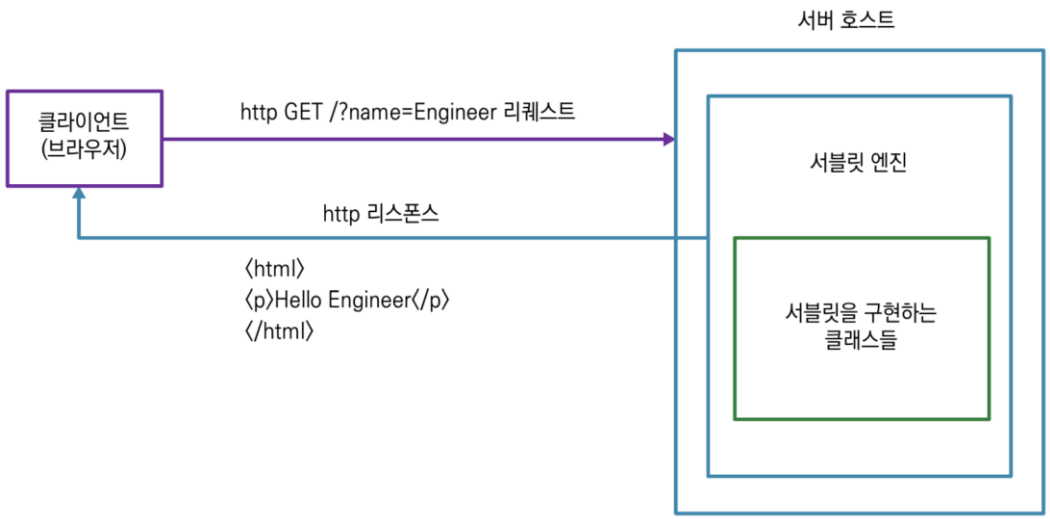

그림 1-11 서블릿 엔진

1.3.7 정리

- 웹 서버
- HTTP
- JSON
- 서블릿엔진
- 스프링부트