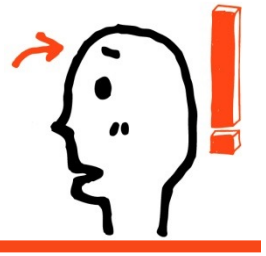
HTTP 이해

1. HTTP Protocol
2. JSON
3. XML

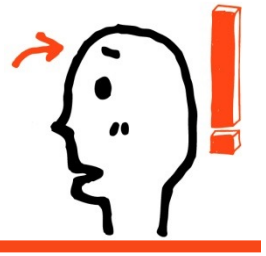
HTTP 프로토콜 알고 넘어가자



HTTP 프로토콜이란?

- **HTTP 프로토콜** : TCP/IP 기반의 네트워크를 바탕으로 WWW(World Wide Web, 웹을 의미한다) 서비스를 위해서 디자인된 통신 규약 즉, 프로토콜(Protocol)
- **웹 서버(server)** : 서버와 브라우저 는 HTTP 프로토콜을 이용해서 데이터(문서)를 주고 받는데 이때 서비스를 제공해주는 것
- **웹 클라이언트(client)** : 서비스를 제공받는 것
- 웹 브라우저는 HTML 문서를 파싱해서 여러분에게 문서 화면을 보여주는 역할

HTTP 프로토콜 알고 넘어가자



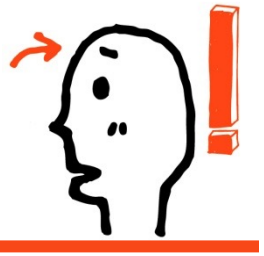
HTTP 프로토콜이란?

HTTP 프로토콜의 특징

1. Stateless 프로토콜

- 서버와 클라이언트 사이에 네트워크 연결을 항상 유지하지 않는 것을 의미
  ↔ Stateful 프로토콜
- 한 번 서버와 커넥션(Connection)을 맺으면
  서버와 커넥션을 유지해서 메시지를 주고 받는 것 (대표 : FTP와 Telnet)
- 요청이 끝나면 커넥션을 끊기 때문에 더 빠르고, 더 많은 클라이언트의 요청 처리 가능
- 세션(Session)이나 쿠키(Cookie) 같은 다양한 우회 방법을 제공
  → 서버와 클라이언트가 메시지를 주고 받을 때 서버가 어떤 클라이언트인지를 인식 가능

HTTP 프로토콜 알고 넘어가자

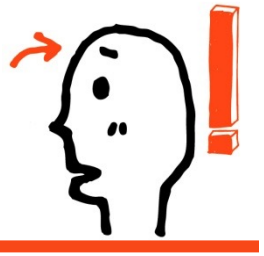


HTTP 프로토콜이란?

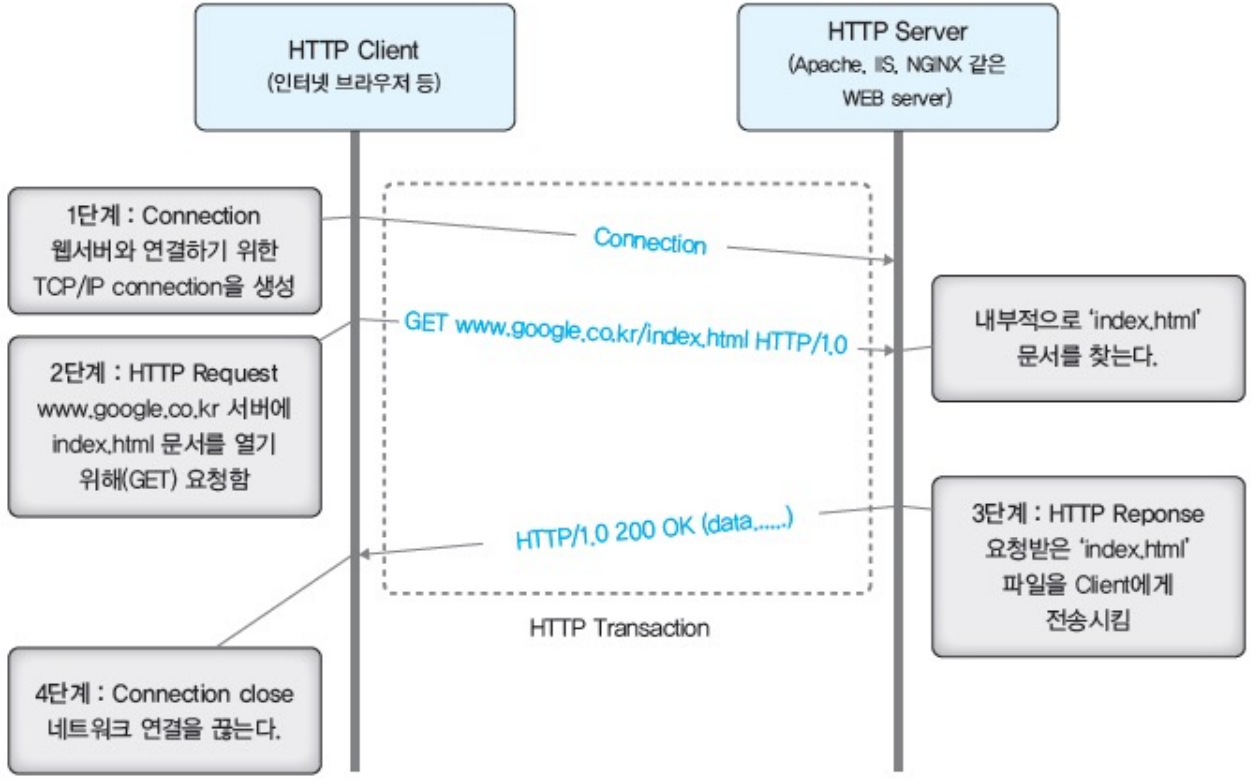
② 트랜잭션 처리

- **트랜잭션** : HTTP 프로토콜에서 클라이언트가 서버로 데이터를 요청하고 서버는 클라이언트에게 데이터를 전송하는 것
- **HTTP 요청 혹은 HTTP Request** : 서버-클라이언트 네트워크 모델에서 클라이언트가 데이터를 전송받기 위해서 웹 서버에게 요청하는 것
- **HTTP 응답 혹은 HTTP Response** : 웹 서버가 그 요청에 대해서 응답하는 것
- 웹 서버는 동시에 여러 개의 요청(Request)을 내부적으로 처리하고 응답
- HTTP 트랜잭션은 클라이언트가 HTTP Request를 서버로 전송할 때 초기화
- 서버는 HTTP Response를 클라이언트로 전송하고 하나의 트랜잭션은 종료

HTTP 프로토콜 알고 넘어가자

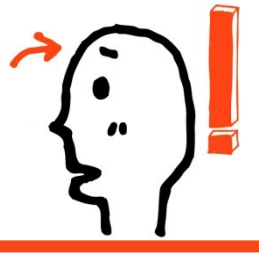


HTTP 프로토콜의 4단계 동작 과정



△ 그림 21-1 HTTP 프로토콜의 동작 과정

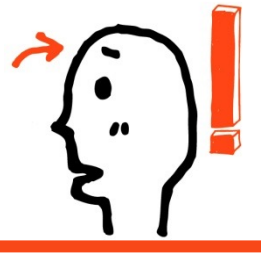
HTTP 프로토콜 알고 넘어가자



HTTP 프로토콜의 4단계 동작 과정

- HTTP 프로토콜 1단계 : **커넥션 생성**
  - 웹 브라우저와 같은 HTTP 클라이언트가 HTTP 서버에게 HTTP Request를 전송하기 위해 먼저 커넥션(Connection)을 맺음
- HTTP 프로토콜 2단계 : **HTTP Request 전송**
  - HTTP 프로토콜에 정의된 명령어들(Method) 중 하나인 GET 명령어를 사용해서 HTTP Request를 전송

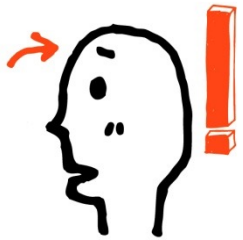
HTTP 프로토콜 알고 넘어가자



HTTP 프로토콜의 4단계 동작 과정

- HTTP 프로토콜 3단계 : **HTTP Response 전송**
  - 세 번째 단계는 서버가 클라이언트에게 HTTP Response 메시지를 전송하는 과정
  - 2단계의 HTTP Request를 수신한 HTTP 웹 서버는
    내부적으로 'index.html'이라는 문서를 찾아서 그 문서를 HTTP 클라이언트로 전송
  - 이때 HTTP 프로토콜에 정의된 상태 코드(status code)라는 값을 같이 전송
- HTTP 프로토콜 4단계 : **커넥션 종료**
  - 마지막으로 HTTP Response를 받은 클라이언트는 1단계에서 생성한 커넥션을 종료

HTTP 프로토콜 알고 넘어가자



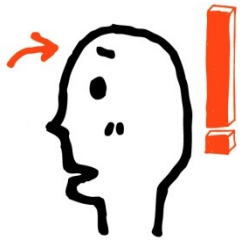
요청-응답 메시지와 상태 코드

- 클라이언트는 서버에게 HTTP Request 메시지를 전달할 때
  반드시 명령어(Method)를 포함해서 전달해야 함

- HTTP Request를 받은 서버는 명령어를 확인하고, 명령어에 따라서 작업을 수행

- 웹 서버가 클라이언트에게 문서를 전달하는 것은
  HTTP 명령어에 따라 처리하는 일 중 가장 대표적인 예

HTTP 프로토콜 알고 넘어가자

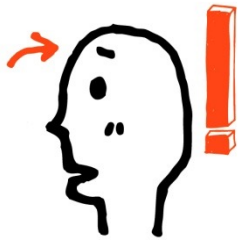


요청-응답 메시지와 상태 코드

명령어(Method)	설명
OPTIONS	웹서버에서 사용할 수 있는 명령어의 종류를 요청한다.
HEAD	HTTP 프로토콜의 헤더 정보만 수신한다.
GET	서버에서 문서를 받아온다. 클라이언트가 서버로 데이터를 전송할 때 URL에 붙여서 전송하며, 보통 URL 뒤에 물음표(?) 기호로 URL과 데이터를 구분한다. 여러 개의 데이터를 전송할 때 데이터 구분은 '&' 기호로 한다. 하지만 GET 명령어에서 전송할 수 있는 데이터의 양은 제한적이며 최대 2048byte까지 전송 가능하다.
POST	서버에서 문서를 받아온다. GET 방식과 다르게 HTTP Request 메시지의 바디에 데이터를 넣어서 전송한다. 보낼 수 있는 데이터의 양은 무제한에 가깝다.

△ 표 21-1 HTTP 프로토콜 표준에 정의된 명령어

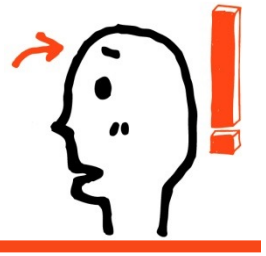
HTTP 프로토콜 알고 넘어가자



요청-응답 메시지와 상태 코드

PUT	서버에 파일을 올리는 명령어로 보통 웹 서버에서는 비활성화되어 있는 기능이다.
TRACE	클라이언트가 전송한 HTTP Request를 그대로 반환한다. 디버깅의 목적으로 사용한다.
DELETE	서버에 파일을 삭제하는 명령어로 보통 웹 서버에서는 비활성화 되어 있는 기능이다.
CONNECT	터널링 방식의 연결을 요청할 때 사용한다. 클라이언트가 HTTP 통신을 하기 위해서 초기에 맺는 커넥션과는 상관 없다.

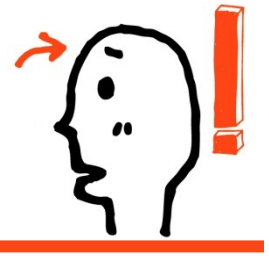
HTTP 프로토콜 알고 넘어가자



요청-응답 메시지와 상태 코드

- **GET 명령어** : 클라이언트는 URL에 데이터를 표기해서 전달
- **POST 명령어** : 클라이언트는 데이터를 HTTP Request 메시지에 숨겨서 전달
- HTTP 클라이언트와 HTTP 서버가 주고 받게 되는 메시지는
  HTTP Request와 HTTP Response 두 가지가 있으며 둘다 헤더와 바디로 구분
- 헤더는 HTTP 프로토콜의 정보를 담고 있는 부분
- 바디는 HTTP를 통해 전송할 메시지나 데이터를 담고 있는 부분
- POST 명령어를 사용 - 데이터는 모두 바디 영역에 포함
- GET 방식 - 바디는 비어 있는 상태로 서버에 전달

HTTP 프로토콜 알고 넘어가자



요청-응답 메시지와 상태 코드

HTTP Request

Header
```
GET http: //www.google.co.kr/indx.html  HTTP/1.1
Host: www.google.co.kr
Accept: text/html, text/plain
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0) Gecko/20100101 Firefox/16.0
```

Body
```
GET의 경우에는 Body에 데이터가 없다.
```

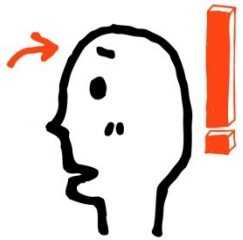
HTTP Response

Header
```
HTTP/1.1 200 OK
Server: Apache
Content-Type:text/html; charset=UTF-8
Date: Sun, 18 Nov 2012 05:31:25 GMT
```

Body
```
<!doctype html> <html itemscope="itemscope" itemtype:"http://schema.org/
WebPage"> <head> <meta itemprop="image" content="/images/.....
```

△ 그림 21-2 HTTP Request와 HTTP Response의 구조

HTTP 프로토콜 알고 넘어가자



요청-응답 메시지와 상태 코드

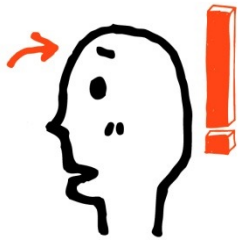
- HTTP Request 메시지의 헤더 부분을 보면 다음과 같은 주요 정보가 포함

```
GET http://www.google.co.kr/indx.html  HTTP/1.1
Host:www.google.co.kr
```

- 헤더의 첫 번째 줄은 "HTTP 프로토콜의 GET 명령어를 이용하여 www.google.co.kr/index.html 문서를 요청하고 HTTP 프로토콜의 버전은 1.1을 사용한다"를 의미

- 두 번째 줄은 'index. html' 문서를 요청할 웹 서버의 주소는 'www.google.co.kr'임을 의미

HTTP 프로토콜 알고 넘어가자



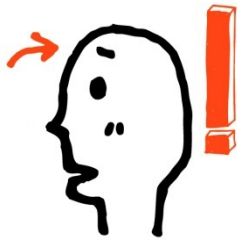
요청-응답 메시지와 상태 코드

- HTTP Response 메시지 헤더에서 가장 중요한 정보

```
HTTP/1.1 200 OK
Content-Type:text/html; charset=UTF-8
```

  - 'HTTP/1.1'은 프로토콜의 종류와 버전을 의미
  - '200 OK'는 클라이언트가 전송한 HTTP Request 메시지에 대한 서버의 처리 결과 코드와 메시지를 의미
  - 숫자 200을 상태 코드(Status Code)
  - 'Content-Type:text/html;'은 HTTP Response 메시지 바디에 포함된 데이터 종류를 알려줌

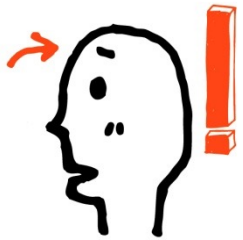
HTTP 프로토콜 알고 넘어가자



요청-응답 메시지와 상태 코드

- 100번대 코드들은 HTTP 정보와 관련된 상태 코드들
- 200번대 코드들은 HTTP 트랜잭션이 성공적임을 의미하는 상태 코드들
- 300번대 코드들은 트랜잭션 을 다른 곳으로 재전송한다는 의미
- 400번대 코드들은 클라이언트에 의해서 발생된 오류를 의미
- 500번대 코드들은 서버에 의해서 발생된 오류들을 의미
- 400~5XX 사이에 있는 상태 코드들은 '에러'라고 판단
- 200번대의 코드는 '성공'이라고 판단해도 무방

HTTP 프로토콜 알고 넘어가자

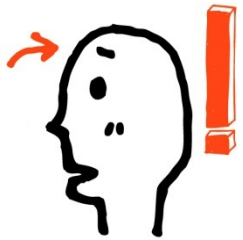


요청-응답 메시지와 상태 코드

상태 코드	설명
100	[Continue] 서버에서는 클라이언트로부터 HTTP Request를 받았음을 의미한다. 그래서 클라이언트로 하여금 계속해서 HTTP Request를 보내라는 의미다.
200	[OK] 클라이언트에서 전송한 HTTP Reuqest가 성공적으로 처리되었음을 의미한다.
201	[Created] PUT 메소드에 의해서 서버에 파일이 정상적으로 생성되었음을 의미한다.
202	[Accepted] HTTP 서버가 정상적으로 HTTP Method를 수신하였음을 의미한다.
204	[No content] 클라이언트의 Request를 정상 수신했지만 서버에서 클라이언트로 전송할 데이터가 없음을 의미한다.

△ 표 21-2 HTTP 프로토콜의 상태 코드 중 일부 내용

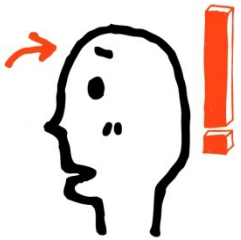
HTTP 프로토콜 알고 넘어가자



요청-응답 메시지와 상태 코드

301	[Moved permanently] 요청한 URL 정보를 다른 URL에 요청한다. 요청한 URL은 영구적으로 다른 URL로 옮겨졌음을 의미한다.
302	[Moved temporarily] 요청한 URL 정보를 다른 URL에 요청한다. 요청한 URL은 실제로 다른 URL에 위치하고 있음을 의미한다.
400	[Bed request] 클라이언트에서 전송한 HTTP Request의 문법이 잘못되어 서버에서 처리할 수 없음을 의미한다.
401	[Unauthorized] 인증이 필요한 페이지를 요청한 경우 HTTP Request 헤더에 Authorization 필드가 비어 있어 인증을 할 수 없음을 의미한다.

HTTP 프로토콜 알고 넘어가자



요청-응답 메시지와 상태 코드

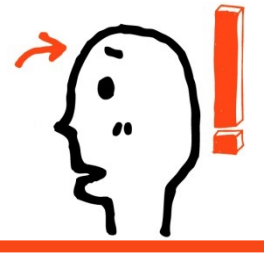
403	[Forbidden] 접근이 금지된 데이터를 클라이언트가 요청한 경우 발생한다.
404	[Not found] 클라이언트가 요청한 페이지가 없을 때 발생한다.
408	[Request timeout] 요청 시간이 지났음을 의미한다.
500	[Internal server error] 내부 서버의 오류에 의해서 정상적으로 데이터를 전송할 수 없음을 의미한다.
503	[Service unavailable] 서버의 문제로 인하여 HTTP 서비스를 할 수 없음을 의미한다. 대부분 HTTP 서버의 과부하 때문에 발생하는 에러다.
505	[HTTP Version not supported] 클라이언트에서 전송하는 HTTP Request의 HTTP 버전을 서버가 지원하지 못할 때 서버가 내려주는 상태 코드다.

20장

JSON 핸들링

셋째 마당 | 자바 프로젝트에서 자주 사용하는 라이브러리 익히기

JSON 핸들링



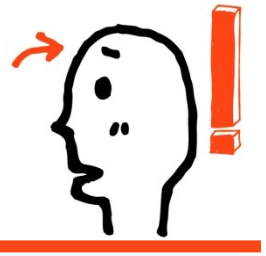
01 JSON 기본 문법 익히기

02 Json-simple 라이브러리 활용하기

03 jSONMapper 클래스 예제 실습하기

01

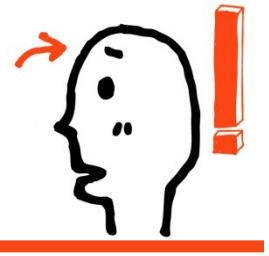
JSON 기본 문법 익히기



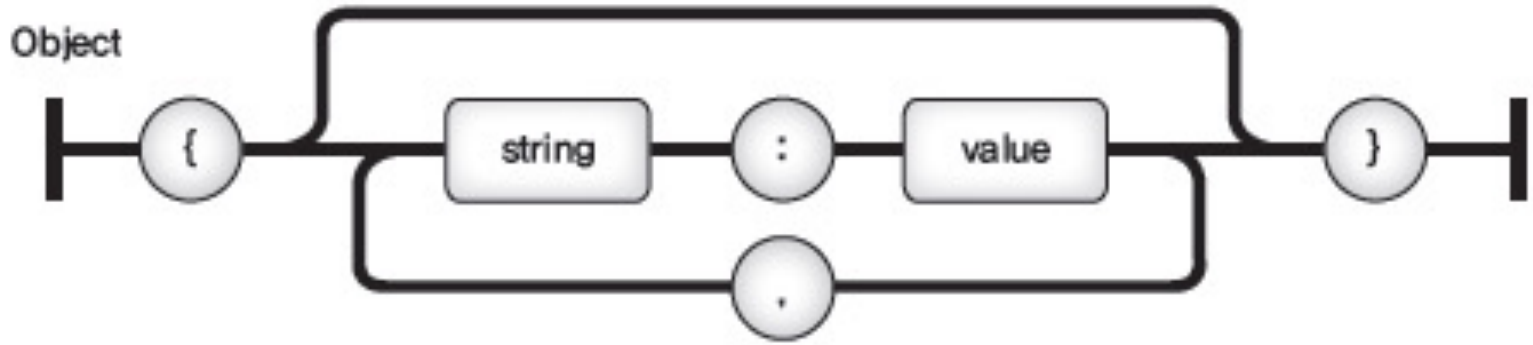
▷ **JSON을 사용함으로써 얻을 수 있는 이점**

1. 데이터 크기가 작다
   - ▷ XML에 사용되던 태그가 없기 때문에 전체 텍스트의 사이즈가 줄어들었음
2. 사람이 읽고 쓰기에 용이하다
   - ▷ 괄호와 세미콜론 같은 간단한 기호로 구성
   - ▷ 데이터 형이 바이너리가 아니라 순수 텍스트(Plain text)
3. 개발 언어와 OS 종류에 구애받지 않는다
   - ▷ JSON은 순수 텍스트로 구성되어 있고 데이터 교환을 위한 표준 표기 방법
   - ▷ 이기종 시스템 간에 데이터 교환 방식

JSON 기본 문법 익히기

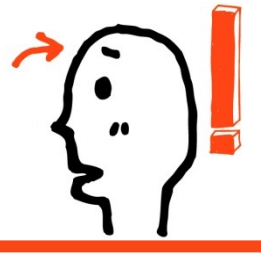


JSON 객체



△ 그림 20-1 JSON 객체를 표현하는 방법을 도식화한 그림

JSON 기본 문법 익히기

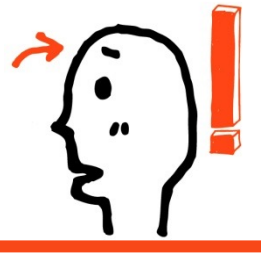


JSON 객체

JSON 규칙

- JSON 객체의 시작과 끝은 중괄호 ({ })로 표기한다
- JSON 객체의 속성은 Name/Value 형태다
- 속성 Name/Value는 콜론(:)을 기준으로 구분한다

JSON 기본 문법 익히기



JSON 객체

JSON 데이터 표현 방법

```
사용예: {“NAME” : “Benjamin Kim”}
        {“URL” : “www.facebook.com”}
```

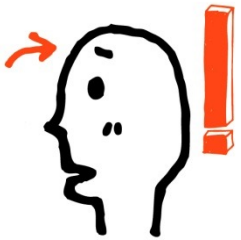
- JSON 객체의 속성은 콤마를 사용해서 구분한다

```
사용예: { “NAME” : “Benjamin Kim”, “URL” : “www.facebook.com”}
```

- JSON 객체의 속성 name은 객체 내에서 유일해야 된다
- String형 데이터는 쌍따옴표를 사용한다

```
사용예: case 1 : {“name” : “BenjaminKim”}
        case 2 : {“age” : 33 }
        case 3 : {“isMale” : true }
        case 4 : {“name” : “Benjamin Kim” , “age” : 33 , “isMale” : true }
```

JSON 기본 문법 익히기



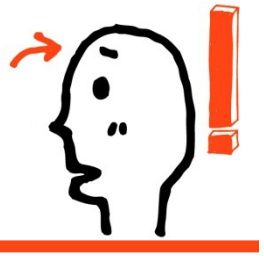
JSON 객체

- JSON 문법에서 객체는 자바의 Hashtable 객체와 매우 비슷한 원리로 데이터를 저장
- 다음과 같이 Hashtable 객체를 생성하고 데이터를 화면에 출력한다고 가정

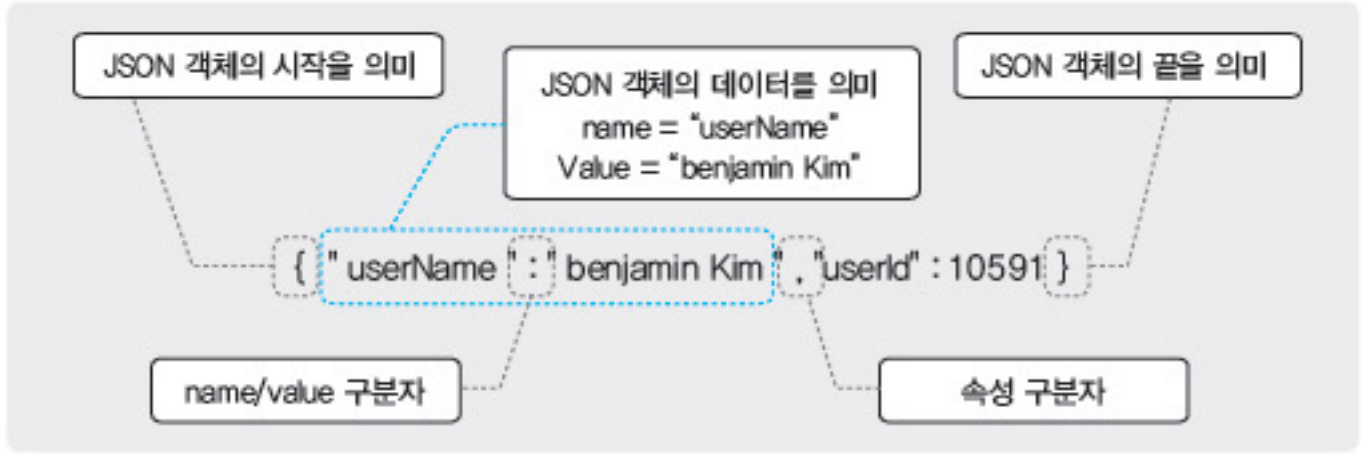
```
Hashtable userInfo = new Hashtable();
userInfo.put("userName", "benjamin Kim");
userInfo.put("userId", 10591);

System.out.println(userInfo.get("userName"));
System.out.println(userInfo.get("userId"));
```

- JSON을 사용한 데이터 표기법이 매우 간단하면서 XML보다 데이터 크기도 줄일 수 있는 것을 확인

JSON 기본 문법 익히기

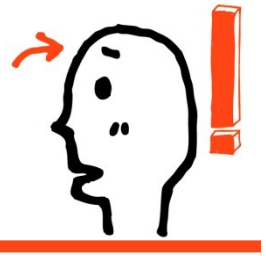


JSON 객체

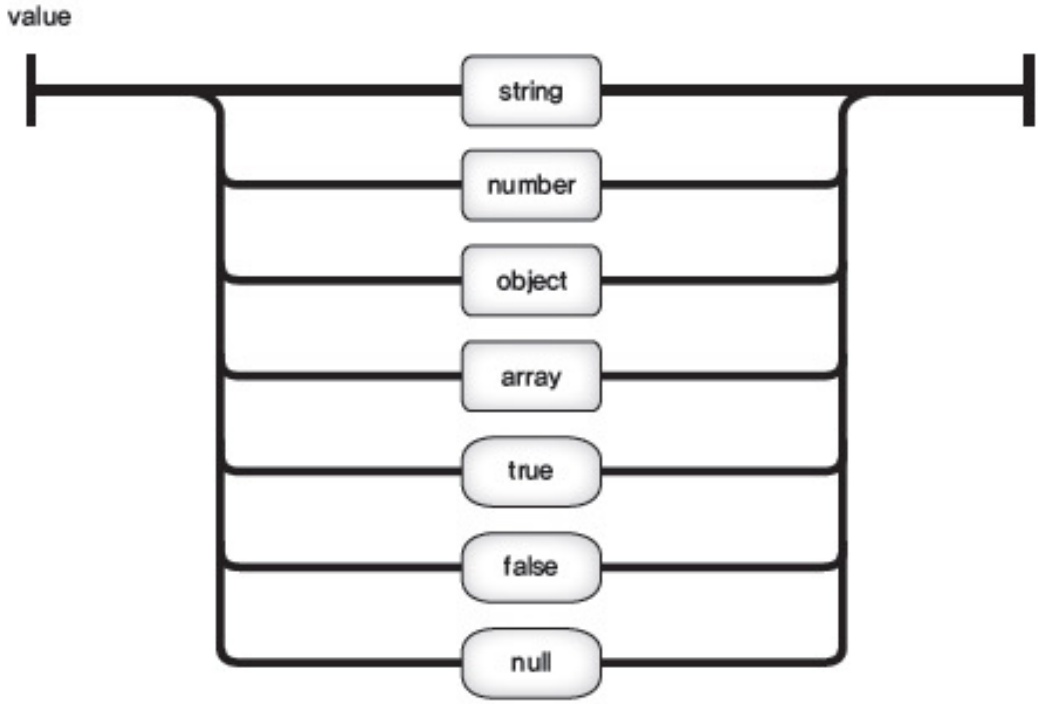


△ 그림 20-2 Hashtable의 데이터를 json으로 표현한 데이터

JSON 기본 문법 익히기

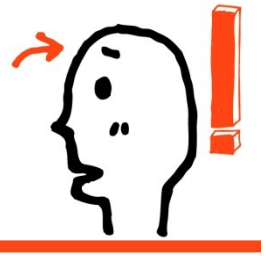


JSON 객체



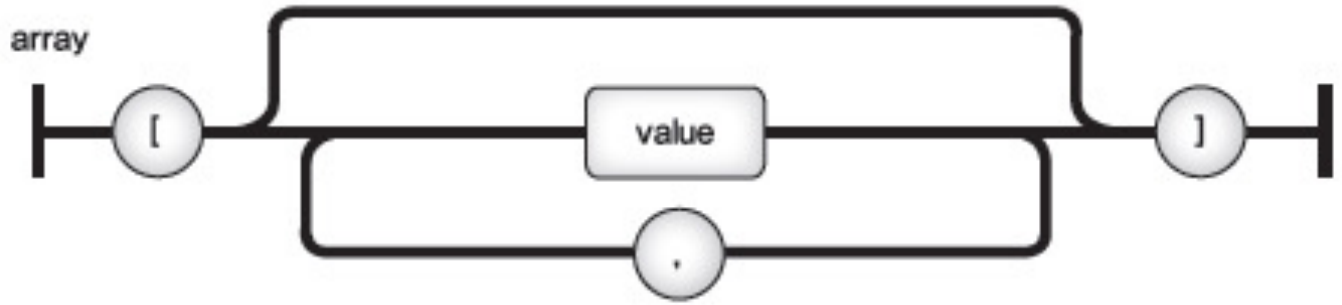
△ 그림 20-3 JSON 객체에 저장 가능한 데이터형

JSON 기본 문법 익히기



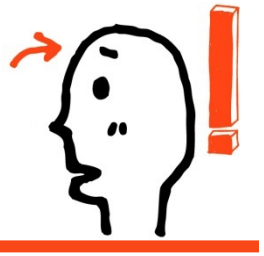
JSON 배열

▷ 여러 개의 데이터를 하나로 표현하기 위해서 JSON 배열(JSON Array) 을 사용



△ 그림 20-4 JSON 배열에 저장 가능한 데이터형

JSON 기본 문법 익히기



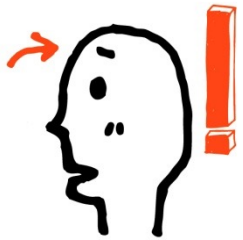
JSON 배열

JSON 배열을 만드는 표기법

- JSON 배열의 시작과 끝은 대괄호를 사용
- JSON 배열의 멤버 변수들은 콤마를 사용해서 구분
- JSON 배열의 구성 아이템으로 JSON 객체도 넣을 수 있음
- JSON 객체의 value 부분에는 JSON 객체를 넣을 수 있음

```
사용예: [ "Benjamin Kim", "Hamzani Ali", "Lodoss" ]
        [ 1, 2, 3, 5, 8, 13, 21]
        [ { "name": "Benjamin" , "age" : 33 } , { "name" : "Hamzani" , "age" : 30 } ]
```

JSON 기본 문법 익히기

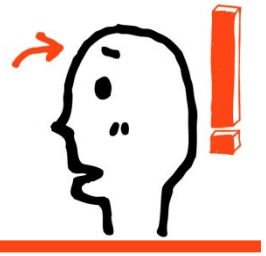


JSON 배열

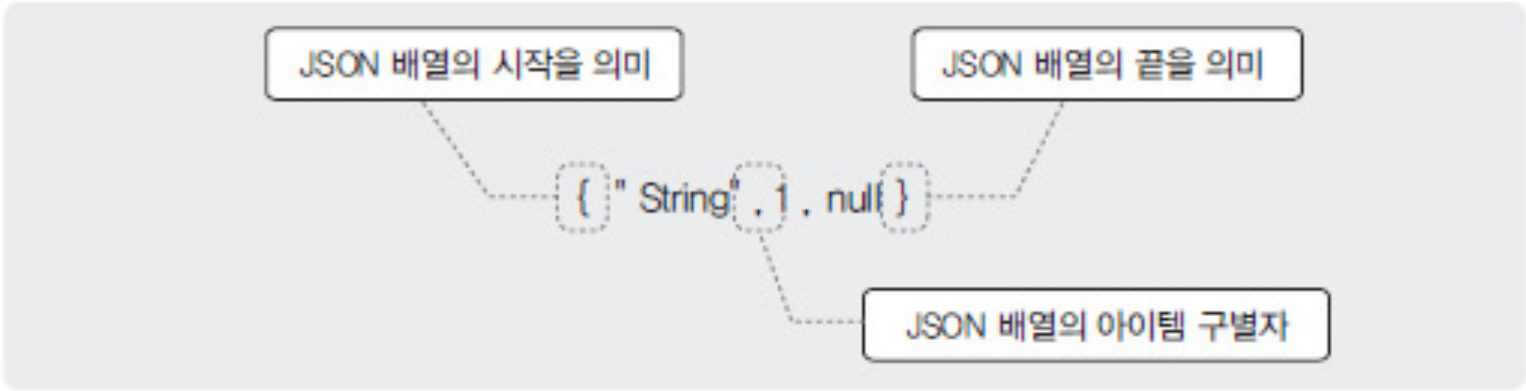
- JSON 배열은 자바의 ArrayList와 비슷한 형태

```
ArrayList vluList = new ArrayList<Object>();
vluList.add("String");
vluList.add(1);
vluList.add(null);

System.out.println("First Value\t: " + vluList.get(0));
System.out.println("Second Value\t: " + vluList.get(1));
System.out.println("Third Value\t: " + vluList.get(2));
```

JSON 기본 문법 익히기

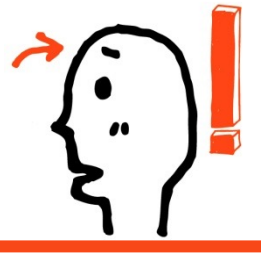


JSON 배열



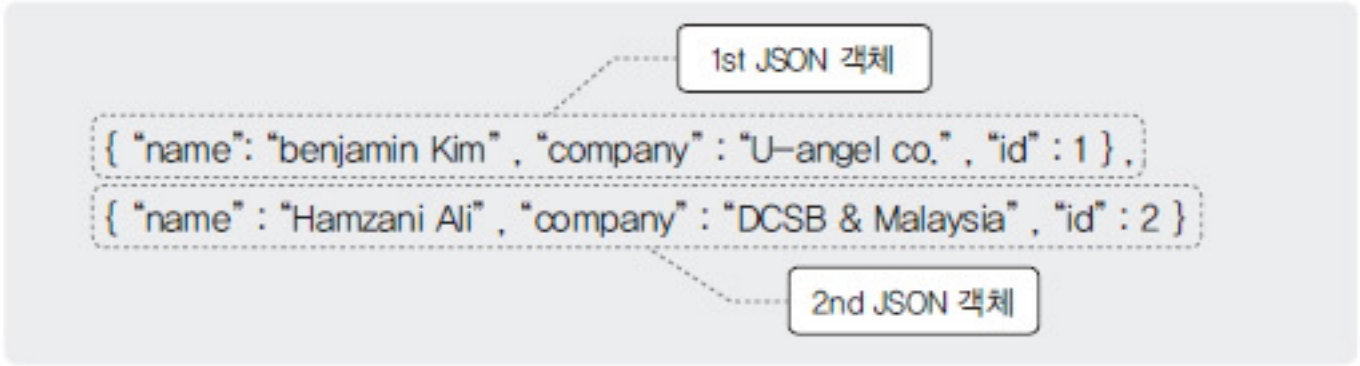
△ 그림 20-5 ArrayList의 내용을 JSON 배열로 바꾼 것

JSON 기본 문법 익히기



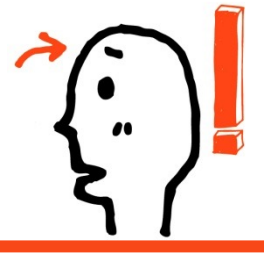
JSON 배열

- JSON 객체와 JSON 배열을 혼합하여 데이터를 구성
- XML과 JSON의 문자량(파일 크기)이 다름

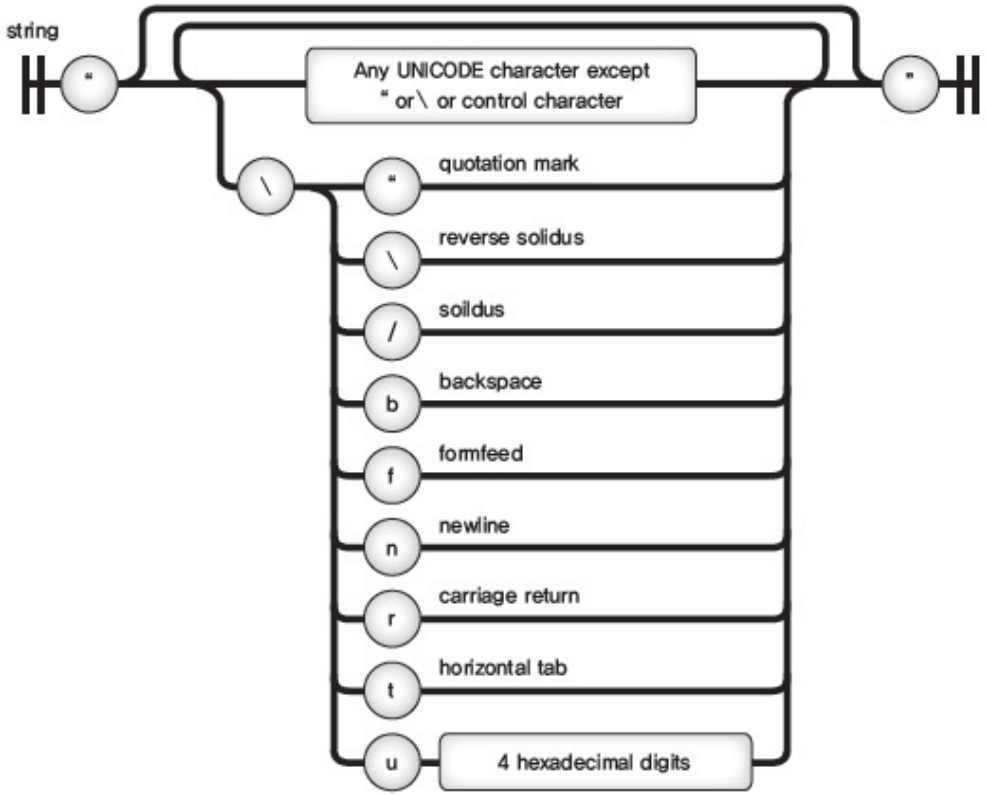


△ 그림 20-6 JSON 배열의 예

JSON 기본 문법 익히기

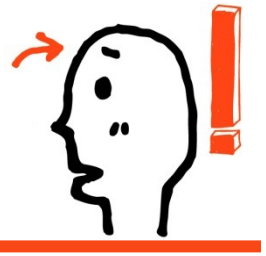


JSON 배열



△ 그림 20-7 JSON 표기법에서 특수 문자 표현

JSON 기본 문법 익히기



JSON 배열

문자	표기법
쌍따옴표 "	"\""
슬래시 /	"\/"
다음 줄로 이동	"\f"
라인 맨 앞으로 이동	"\r"
유니코드	"\u[16진수 유니코드]"
역슬래시 \	"\\"
백스페이스	"\b"
새로운 라인	"\n"
탭	"\t"

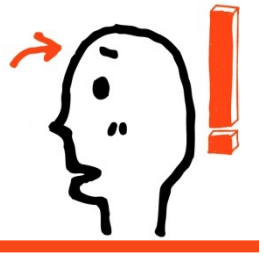
△ 표 20-1 JSON의 특수 문자 표기법

19장

XML 핸들링

셋째 마당 | 자바 프로젝트에서 자주 사용하는 라이브러리 익히기

XML 핸들링



01 XML에 대한 개요

02 DOM XML 파서는 언제 사용할까?

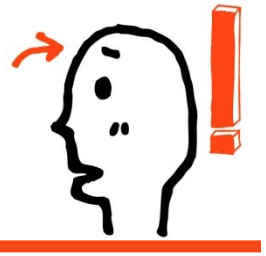
03 DOM XML 파서 다루기

04 SAX XML 파서도 기억하자

05 DOM 파서 vs SAX 파서 비교하기

01

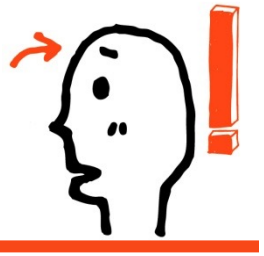
XML에 대한 개요



- **XML(eXtended Markup Language)**
  - 데이터를 저장하거나 구조화하기 위한 일종의 표기 방법
  - HTML을 파싱하여 문서를 화면에 보여주는 방법이 서로 달라 상호 호환성에 문제
  - HTML5가 발표된 현재에도 상호 호환성 문제는 여전히
  - HTML로 문서의 구조나 정보를 표현하는 것에는 한계
  - 이런 단점을 극복하기 위해서 XML 표준 규격이 발표
  - 표준 규격은 다른 OS나 개발 언어 환경에서도 서로 공용(share)할 수 있다는 장점

01

XML에 대한 개요

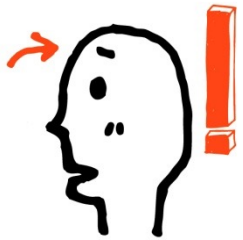


XML이 사용되는 분야

- XML을 활용하면 구조화된 데이터를 저장한 문서를 만들 수 있으며
  보다 자유로운 문서 표현이 가능

1. 데이터 저장과 표현
   - 트리(Tree) 형태로 구조화될 수 있어 프로그램에서 데이터를 읽고 쓰기에 매우 적합
2. 데이터 공유
   - 데이터가 문서에 저장되므로 어떤 프로그램이든지 쉽게 내용을 확인할 수 있기 때문에
     개발 언어 혹은 OS에 매우 독립적인 형식

XML에 대한 개요



XML이 사용되는 분야

3 시스템/프로그램의 유연성 증대

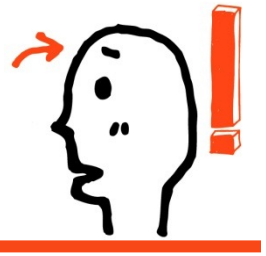
- 대부분의 오픈 소스 라이브러리나 프레임워크에서는 프로그램 외부에서 기능에 대한 정의 및 설정을 XML 파일로 조정할 수 있음

4 자유로운 문법

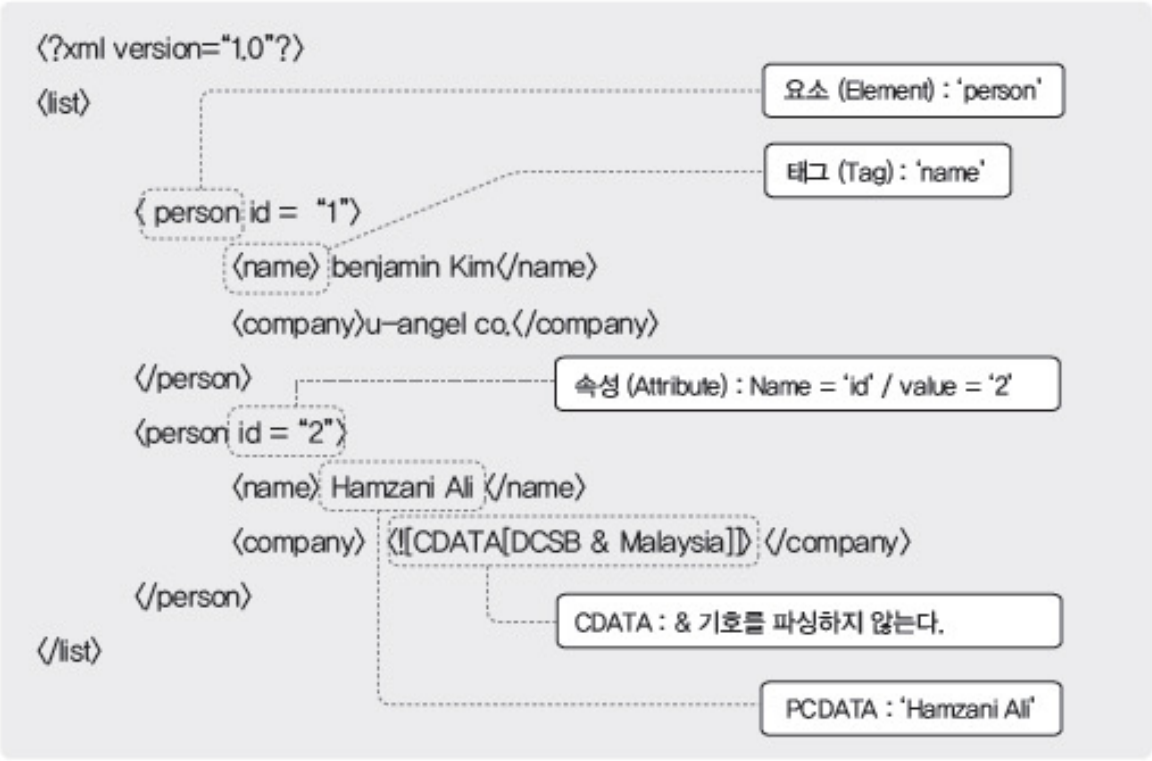
- XML은 태그와 속성을 개발자가 자유롭게 선언할 수 있음

  최소의 문법에 대해서만 정의하고 있으므로 데이터를 표현하는데 매우 자유로움

XML에 대한 개요

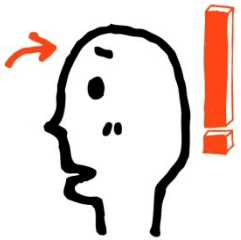


XML을 구성하는 요소들



△ 그림 19-1 XML 구성 요소의 예

XML에 대한 개요

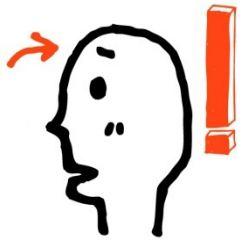


XML을 구성하는 요소들

① 태그(Tag)

- XML 문서는 태그를 이용해서 데이터를 구조화하고 문서를 구성
- **쌍태그** : 태그의 시작과 끝으로 데이터를 감싸고 있는 형태
- **홑태그** : 데이터를 감싸지 않으면서 혼자 존재하는 것
- 홑태그를 사용할 때는 반드시 태그의 끝을 의미하는 슬래시(/) 문자를 선언
- 쌍태그는 슬래시가 없는 시작 태그로 시작해서 슬래시가 포함된 종료 태그로 끝을 표현

XML에 대한 개요

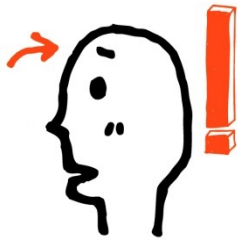


XML을 구성하는 요소들

② 요소(Elements)

- 데이터의 의미를 나타내는 반면 태그는 데이터의 구조를 구성하는 용도로 사용
- 요소는 태그에 포함되는 의미
- 태그 <person id = “1”>의 요소는 person
- <요소이름></요소이름>과 같이 태그를 만들어서 문서를 구성

XML에 대한 개요



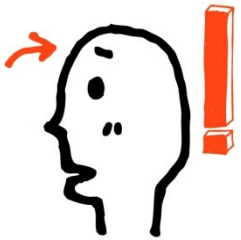
XML을 구성하는 요소들

③ 속성(Attribute)

- 요소에 대한 부가적인 정보를 표현하는데 사용
- 속성은 하나의 요소에 여러 번 사용될 수 있으며 '속성 이름 = 속성값'의 형태로 사용
- 여러 개의 속성이 사용된 person 태그의 예

```
<person id="2" bloodType="O">
<name>Benjamin Kim</name>
</perons>
```

XML에 대한 개요

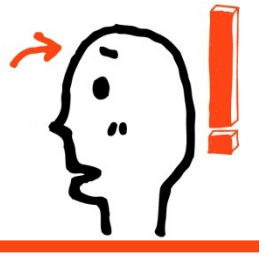


XML을 구성하는 요소들

④ PCDATA(Parsed Character DATA)

- **파싱(parsing)** : XML 문서와 같이 특정 문법이나 규칙에 의해서 만들어진 것을 읽는 것
- **파서(parser)** : 문서를 읽는 프로그램
- PCDATA는 XML 파서에 의해서 파싱될 수 있는 데이터를 의미
- 시작 태그와 종료 태그 사이에 있는 데이터
- 특수 문자가 포함되지 않아 XML 파서가 오류 없이 읽을 수 있는 데이터

XML에 대한 개요



XML을 구성하는 요소들

⑤ CDATA(Character DATA)

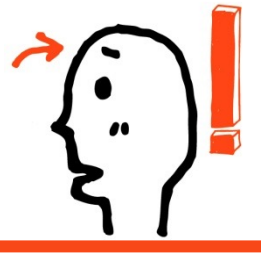
- 특수 문자가 포함된 데이터를 표현하고 싶을 때
- CDATA 영역을 표기하고 특수 문자를 사용
- CDATA 영역을 표기하는 방법

```
사용법 : <![CDATA[      ]]>
사용예 : <![CDATA[ Tea & Coffee ]]>
```

- CDATA 영역에 있는 데이터들에 대해서는 파싱 과정을 수행하지 않음
- CDATA 표기법

```
사용예 : <person><![CDATA[Benjamin & Hamzani]]></person>
```

XML에 대한 개요



XML을 구성하는 요소들

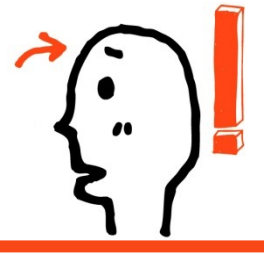
⑥ 엔티티(Entity)

- 'XML을 구성하는 저장 단위'라고 정의
- XML 파서는 미리 정의된 엔티티를 파싱 도중에 만나면 미리 정의된 문자로 치환

엔티티 표현법(entity)	치환 문자(character)
&lt;	<
&gt;	>
&amp;	&
&quot;	" (쌍따옴표)
&apos;	' (홑따옴표)

△ 표 19-1 사전에 정의된 엔티티

XML에 대한 개요



간단한 XML 예제

코드 19-1 personList.xml

```
1   <?xml version="1.0"?>
2   <list>
3      <person id = "1">
4         <name>benjamin Kim</name>
5         <company>u-angel co.</company>
6      </person>
7      <person id = "2">
8         <name>Hamzani Ali</name>
9         <company><![CDATA[DCSB & Malaysia]]></company>
10     </person>
11  </list>
```