**웹 프로그래밍 기초**

**\* 웹 프로그래밍을 위한 프로그램 언어들**

|  |  |
| --- | --- |
| **FORTRAN** | 최초의 고급 언어. 과학 계산용으로 주로 사용 |
| **COBOL** |  |
| **PROLOG** | 논리식을 토대로 오븍트간 관계에 관한 문제 해결에 주로 사용 |
| **C** | 시스템 프로그래밍에 적합한 언어 |
| **Erlang** | 함수형 병행성 언어. 통신 인프라에 사용 |
| **Lisp** | 함수형 언어 |
| **Swift** |  |
| **Kotlin** | InteliJ 사에서 개발한 언어. JVM 기반이며 Java와 상호 운영 100% 지원 |
| **Clojure** | 범용적인 함수형 언어 |
| **Python** | 데이터 과학, 웹사이트 개발, 기계 학습 분야에서 사용. |
| **Java** |  |

**\* 웹의 동작(HTTP 프로토콜 이해)**

인터넷은 네트워크의 네트워크라고 할 수 있다. 네트워크가 모여서 인터넷이 되고 이런 네트워크 간의 통신은 TCP/IP 프로토콜을 활용한다.

웹 브라우저는 HTTP 프로토콜을 통해 통신을 한다. HTTP는 서버와 클라이언트가 인터넷 상에서 데이터를 주고 받기 위한 프로토콜이다. 현재는 계속 발전하여 HTTP/2 까지 버전이 등장했다. 교육에선 가장 많이 사용되는 HTTP v1.1을 사용한다.

네트워크 통신 방법

|  |  |
| --- | --- |
| **유상태 방식** | 접속이 계속 유지된 상태로 클라이언트와 서버 간에 지속적으로 데이터를 주고 받을 수 있음(채팅/온라인 게임에 적합)  하지만 서버가 유지할 수 있는 수는 정해져 있으므로 많은 클라이언트 요청을 처리하려면 서버의 수가 많아야 한다. |
| **무상태 방식** | 필요한 경우에만 연결을 맺고 요청에 대한 응답 처리후 연결을 끊게 된다. HTTP는 무상태 방식을 사용한다. 하나의 서버가 많은 요청, 응답 처리가 가능하다.  불특정 다수를 대상으로 하는 서비스에 적합하다. 하지만 클라이언트의 이전 상태를 알 수 없어 정보를 유지하기 위해 Cookie와 같은 기술 필요. |

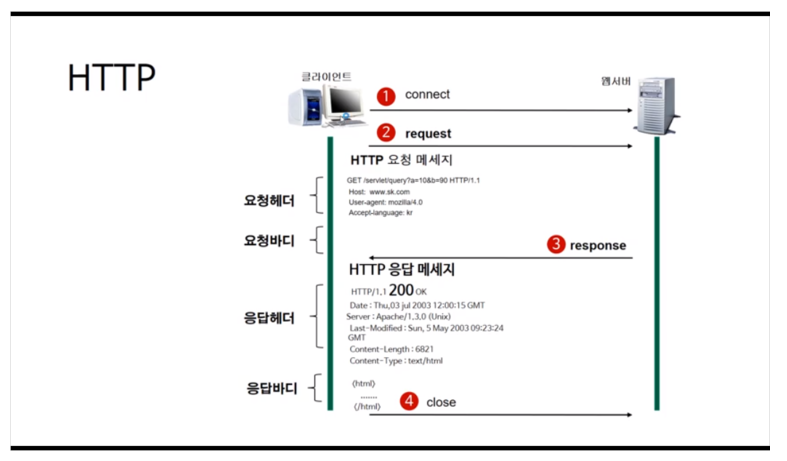
URL (Uniform Resource Locator)

인터넷 상의 자원의 위치를 의미한다. 특 정 웹 서버의 특정 파일에 접근하기 위한 경로 혹은 주소를 의미한다.

URL은 아래와 같이 크게 세 가지로 구성된다.



HTTP의 동작 방식을 간단히 그림으로 살펴보면 아래와 같다.



HTTP 서버/클라이언트가 통신을 하기 위해선 정해진 메시지를 송수신 하게 된다. 상세한 메시지 스펙은 검색을 통해 확인할 것.

**\* Browser의 동작**

참고 사이트 : <https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/>

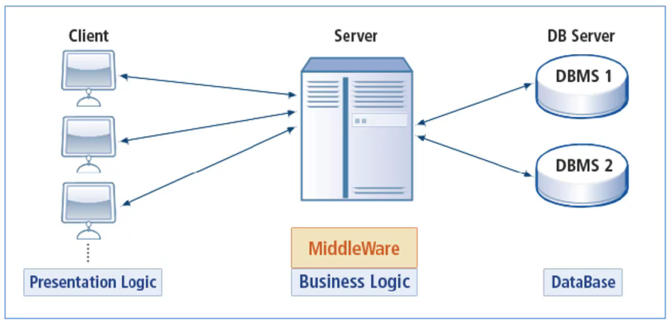
**\* 웹 서버**

웹 서버의 가장 중요한 기능은 클라이언트가 요청하는 HTML 문서나 각종 리소스를 전달하는 것이다. 웹 브라우저가 요청하는 리소스는 서버에 저장되어 있는 정적인 데이터(예를 들어 HTML, CSS, JavaScript와 같이 컴퓨터에 저장된 파일) 또는 동적인 결과(웹 서버에서 실행되는 프로그램에 의하여 만들어진 데이터)일 수 있다.

**\* WAS**

초기 클라이언트/서버 구조와 DBMS 구조에선 클라이언트 소스에서 직접 DB에 접속하여 데이터를 처리 하였다. 하지만 클라이언트 단에 비즈니스 로직이 많아지면서 기능이 추가될 때마다 클라이언트 소스를 배포한다거나 유지보수, 보안 이슈가 나오면서 미들웨어라는 개념이 나오게 되었다.

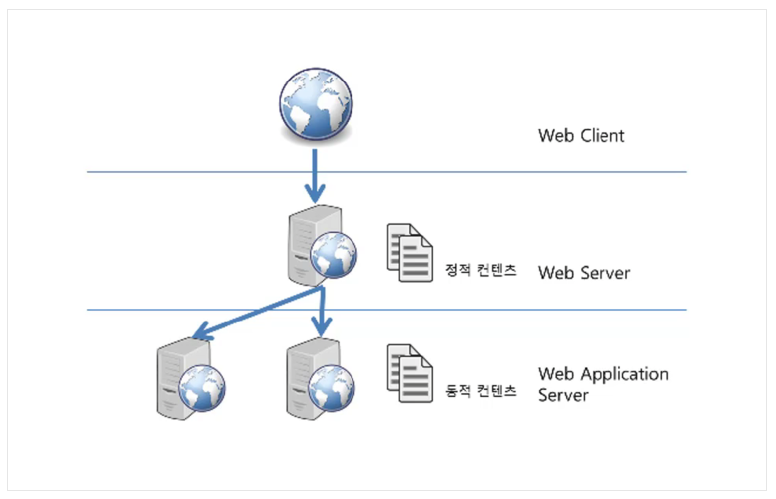
미들웨어는 아래 그림과 같이 클라이언트와 DBMS 사이에 서버를 추가하여 클라이언트에선 더 이상 비즈니스 로직을 수행 안하고 미들웨어에게 요청을 보내면 미들웨어에서 DBMS와 통신하여 클라이언트에게 응답을 할 수 있도록 하는 서버이다.



WAS는 미들웨어의 일종으로 정적인 데이터만을 보여줬던 최초 웹브라우저는 여러가지 동적인 기능(예를 들어 사용자가 입력한 데이터를 기반으로 조회 결과 확인)을 사용자로부터 요구 받았고 이를 위해 웹 서버에 프로그래밍을 추가하게 되었지만(CGI) 이 방식은 단순한 프로그래밍 사용엔 문제가 없었지만 그 기능이 점점 복잡해지면서 WAS가 나오게 되었다.

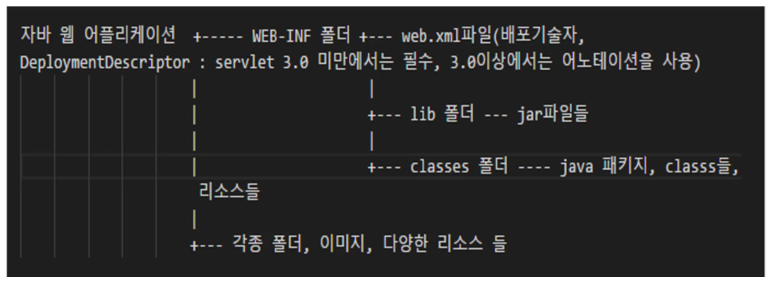
WAS는 크게 프로그램 실행 및 데이터베이스 접속 환경 제공, 트랜잭션 관리, 업무 비즈니스 로직 처리 수행 기능을 제공한다.

일반적으로 WAS도 웹 서버처럼 정적인 데이터를 응답할 수 있다. 하지만 보통 클라이언트 요청이 많은 시스템은 아래와 같이 웹 서버를 별도로 WAS 앞단에 설치하고 동적인 데이터가 있을 때 WAS에게 요청하는 구조를 가지고 있다. 가장 대표적인 이유로 하나의 WAS가 예상치 못한 문제로 Shutdown 된 경우 앞 단의 웹서버가 동적인 데이터를 다른 정상 동작하는 WAS에게 요청할 수 있기 때문이다.



**\* 서블릿이란?**

자바 웹 어플리케이션의 폴더 구조는 다음과 같다. 서블릿 v3.0 부터는 WEB-INF/web.xml 파일을 대신하여 어노테이션을 사용할 수 있다.



서블릿이란 자바 웹 어플리케이션의 동적인 처리를 담당하는 프로그램으로, 일반적으로 서블릿은 JSP와 같이 사용한다. 서블릿의 응답으로 HTML 값을 하나하나 OutputStream에 작성하기 힘들기 때문에 JSP를 같이 사용하기 시작했다.

**\* 서블릿 작성 방법**

@WebServlet(“/exam”) /\*\* URL 매핑 정의 서블릿 v3.0 이상에서 지원 \*/

public class ExamServlet extends HttpServlet {

}

1. 응답을 보낼 땐, 브라우저가 응답 데이터 타입을 알 수 있도록 response.setContentType을 정의해주어야 한다. e.g. setContentType(“text/html;charset=utf-8”)

2. 응답 메시지 바디 작성 시엔 response.getWriter를 통해 작성을 위한 OutputStream을 받는다.

**\* 서블릿 Lifecycle**

WAS는 클라이언트로부터 요청을 받으면 요청에 해당하는 서블릿이 메모리에 있는지 확인 후 서블릿을 생성하게 된다. 로직으로 표현하면 아래와 같다.

if(servlet in memory) {

load servlet class

call init()

}

call service()

즉 WAS 요청 마다 서블릿을 매번 생성하는게 아니라 단 하나의 서블릿만 생성하고 이 후엔 service() 메소드만 호출한다. 이 후 Servlet 클래스가 수정(웹 어플리케이션 갱신)되거나 WAS가 종료되면 destroy() 메소드가 호출된다.

**\* Request, Response 객체 이해하기**

WAS는 요청이 들어오면 Request, Response 객체를 생성하여 웹 어플리케이션에게 전달한다.

