인터넷 프로그래밍

**HTML**

**HTML 이란**

Hyper Text Markup Language의 약자로, **웹사이트를 만드는 가장 기초가 되는 마크업 언어이다**.

**여러 태그들로 구성**되어 있으며 각 태그들을 사용하여 원하는 형태의 문서를 만들 수 있다. (**웹페이지의 내용과 구조를 담당**)

**Hyper Text의 원리를 이용**하여 여러 문서를 연결시켜 다양한 정보를 손쉽게 검색하여 볼 수 있게 만들어 준다.

**\*\* 월드 와이드 웹 (World Wide Web, WWW, W3) \*\***

인터넷에 연결된 컴퓨터를 통해 사람들이 정보를 공유할 수 있는 전 세계적인 정보 공간을 의미

간단하게 웹이라고 부른다.

**\*\* Markup 언어 \*\***

특별한 기호나 표기를 사용하여 글의 서식과 스타일을 정해주는 언어 (프로그래밍 언어 X)

**HTML의 문법**

1. 요소 (Element)

HTML의 요소는 **시작 태그 + 콘텐츠 + 종료 태그**로 구성된다.

**<title> Page title </title>**

1. 시작 태그 (start tag): 어떤 태그를 사용하는지 명시
2. 콘텐츠 (contents): 해당 태그가 적용되기를 바라는 텍스트 문구
3. 종료 태그 (end tag): 해당 태그가 적용되는 끝 지점을 명시
4. 속성 (Attribute)

HTML에게 추가적인 정보나 명령을 받을 수 있게 정보들을 전달해주는 것

**<img src = "html.png"> 🡨이미지 정보**

src : Attribute Name (속성명)

html.png : Attribute Value (속성값)

1. HTML의 Tag
   1. <!DOCTYPE html>: **문서 형식 정의 태그**로 출력할 웹 페이지의 형식을 브라우저에게 전달한다. 문서의 최상위에 위치해야 하며 대소문자를 구별하지 않는다.
   2. <html>: **전체 HTML 문서를 감싸는 태그**, 브라우저에게 HTML 코드가 해당 태그 내부에 존재한다고 알려준다. 하나만 존재해야 하고 HTML바깥에 DOCTPYE을 제외한 다른 태그가 존재하면 안된다.
   3. <head>: **HTML 문서에 대한 정보를 나타내는 부분**으로 주로 외부소스를 참조해야 할 경우 사용한다.
   4. <body>: **HTML 문서에서 실제적으로 보여지는 부분**

**<!DOCTYPE html> 🡨 HTML5 표준을 정의하며 없을 경우 4버전으로 인지한다.**

**<html>**

**<head>**

**화면에 보여지기 위한 추가적인 정보(메타정보)를 담고 있는 부분**

**</head>**

**<body>**

**화면에 보여지는 부분**

**</body>**

**</html>**

1. 상대경로와 절대경로
   1. 상대경로 (Relative): 현재 위치를 기준으로 해당 자료의 위치를 표현한다. 주로 웹사이트 내부의 자료를 연결할 때 사용
   2. 절대경로 (Absolute): 전체 인터넷을 기준으로 해당 자료의 위치를 표현한다. 주로 외부 웹페이지로 연결할 때 사용

Ex) index.html문서 연결

상대경로: index.html 절대경로: <https://google.com/index.html>

**HTML5**

HTML의 새로운 버전으로 Client Side Technology 기술의 중심이 되는 마크업 언어로 **모바일 웹에서의 핵심 기술**

**차세대 웹 표준**으로 기존 텍스트와 하이퍼링크만 표시하던 HTML이 멀티미디어 등 다양한 애플리케이션까지 표현, 제공하도록 진화한 **“웹 프로그래밍 언어”**

오디오, 비디오, 그래픽 처리, 위치정보 제공 등 다양한 기능을 제공함으로써, **웹 자체에서 처리할 수 있는 기능이 대폭 향상됨**

HTML5의 등장과 그 배경 설명 🡺 <https://webclub.tistory.com/491> 에서 Info부분

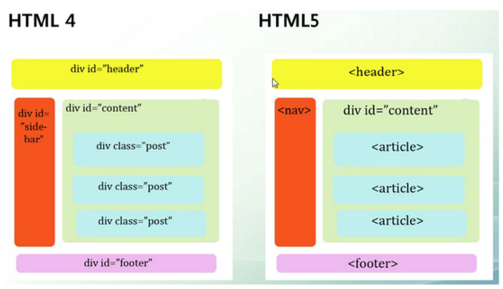
특징

1. 클라이언트와 서버와의 통신이 가능 🡺 GET방식과 POST방식이 존재
2. 다른 외부 Active-X(Internet Explorer용 플러그인)와 같은 plug-ins을 사용하지 않고도 웹 서비스 제공할 수 있을 정도로 많은 기능이 추가됨
3. **시맨틱(Semantic) 태그를 사용한 레이아웃 구성 (새로 추가됨)**

\*\* Semantic Tag \*\*

웹페이지의 구조를 쉽게 이해할 수 있도록 정의된 태그를 의미한다. 즉, 태그의 이름을 통해 문서의 구조를 짐작할 수 있고 레이아웃을 구성하기 위해서 사용한다. (컨테이너로의 역할)

HTML5이전에는 <div>와 <span>에 id와 class속성을 이용하여 구역을 나누고 스타일을 지정했지만 HTML5에서는 시맨틱태그를 이용하여 코드의 가독성을 높이고 명확하게 한다. 즉, <div class=”header”> 🡪 <header>와 같이 뜻이 명확한 태그를 제공



GET vs POST 방식

1. GET: **클라이언트에서 서버로** 어떠한 리소스로부터 **정보를 요청하기 위해 사용되는 방식**으로 게시판의 게시물을 조회할 때 쓸 수 있다. (DB로 따지면 SELECT에 가까움)

Get을 통한 요청은 URL 주소 끝에 파라미터로 포함되어 전송(query string)되고 **?**로 구분한다. (**요청 패킷의 헤더를 통해 전달**)

www.example.com/show?name1=value1&name2=value2

특징

1. 캐시가 될 수 있고 데이터가 북마크 될 수 있다.
2. 전송 데이터 크기가 작고 내용이 URL에 노출되기 때문에 중요한 정보를 다루면 안된다. (보안 안됨)
3. POST: **클라이언트에서 서버로** **리소스를 생성하거나 업데이트하기 위해 데이터를 보낼 때 사용되는 방식**으로 게시판에 게시글을 작성하는 작업 등을 할 때 사용할 수 있다. (DB로 따지면 CREATE에 가까움 🡪 파일 전송, 로그인 전송)

전송할 데이터를 **HTTP메시지 body부분에 포함시켜서 전송**한다.

[www.example.com/show HTTP/1.1](http://www.example.com/show%20HTTP/1.1)

Host: w3schools.com

name1=value1&name2=value2

특징

1. 캐시 되지 않고 데이터가 북마크 되지 않는다.
2. 전송 데이터의 길이 제한이 없고 내용이 URL에 노출되지 않음(보안 유지, 암호화되어 보안성을 보장하는 것은 아님) 🡺 전송데이터 크기가 크거나 주소 표시줄에 데이터 내용을 감주고자 할 때 사용

**CSS (Cascading Style Sheets)**

**웹 페이지 구성 요소의 스타일을 정의하는 언어** (검색창의 너비, 버튼의 크기, 구성요소 배치 등)

장점: 스타일 변경시 스타일 정보만을 바꾸어 주면 되므로 **변경이 쉽고 일관성 유지가 용이**

**HTML에 적용하는 방법 (선언 위치)**

1. INLINE

HTML의 특정 태그에 직접 style을 적용하는 것

1. HTML 내부에 stylesheet 작성

<style>태그를 이용하여 적용하는 방법으로 <head>태그 안에 작정한다. 태그를 선택하는 규칙에 따라 일괄 적용된다.

1. HTML 외부에 stylesheet 작성

<link>태그를 이용하며 css 확장자로 저장된 stylesheet 파일을 href 속성을 이용해 삽입한다.

\_index.html\_

<head>

<title>Page title</title>

**<Link rel= "stylesheet" type = "text/css" href= "myFirstStyle.css"/>**

**<style>**

**h1 {**

**color: red;**

**font-style: italic;**

**}**

**</style>**

</head>

<body>

<h1 **style= "color: red; font-style: italic"**>인라인 style<h1>

</body>

\_**myFirstStyle.css\_**

**h1 {**

**color : red;**

**font-style: italic;**

**}**

1. 스타일 적용 우선순위: 하나의 요소에 대해 외부, 내부, 인라인 스타일이 서로 다르게 지정되고 있다면 우선순위 규칙에 의해 적용
   1. 기본적으로 최근에 선언된 스타일로 적용
   2. 속성값 뒤의 !important > 인라인 스타일 > id > class > tag 요소
   3. 우선순위가 같다면 개수가 많은 css가 우선순위 높다.

**CSS Selector**

css에서 요소를 선택하는 규칙으로 보통은 tag단위로 style이 적용된다.

**Tag별 다른 style을 적용하고 싶은 경우**

1. 각각의 요소에 고유한 **ID속성을 이용**: 자유롭게 이름 부여 가능, 문서 내에서 요소별 유일한 값으로 특정 요소의 이름 붙이는 것에 사용됨
2. Class를 만들고 요소에 **class를 부여**: 자유롭게 이름 부여 가능, 여러 요소가 동일 class사용 가능하고 각 요소는 여러 class 가질 수 있다. 주로 스타일 분류에 사용됨

**자바스크립트 (JavaScript)**

**객체(object)기반**의 스크립트 언어로 **웹의 동작을 구현할 수 있다**.

현재 컴퓨터나 스마트폰 등에 포함된 대부분의 웹 브라우저에는 자바스크립트 인터프리터가 내장되어 있다.

특징

1. 서버가 아닌 클라이언트에서 실행된다. (client side script, 웹 브라우저에서 실행되는 언어)
2. 동적이며, 타입을 명시할 필요가 없는 인터프리터 언어이다. (C언어와 같은 언어가 소스파일 작성 후 컴파일을 통해 사용자가 실행할 수 있는 실행파일을 만들어 사용하는 반면, **인터프리터 언어는 컴파일 작업을 거치지 않고 소스코드를 바로 실행할 수 있는 언어**를 의미)
3. 객체 지향형 프로그래밍과 함수형 프로그래밍을 모두 표현할 수 있다.

**자바 vs 자바스크립트** 🡺 이름 유사하지만 **전혀 다른 언어**

|  |  |  |
| --- | --- | --- |
| 항목 | 자바스크립트 | 자바 |
| 실행방법 | 브라우저에 의해 해석됨 (스크립트 언어) | 서버에서 컴파일 된 Byte Code를 클라이언트에서 해석됨 (컴파일 언어) |
| 언어성격 | Object-Based Language  클래스 미지원  프로토타입 기반 언어 | Object-Oriented Language  클래스 지원  상속과 다형성 지원 |
| 사용형태 | HTML문서 내에 소스코드 기술 (HTML과 결합되어 사용) | HTML문서에 작은 공간을 얻어 수행 (HTML과 별도 존재 가능) |
| 변수형 정의 | 미리 정의할 필요 없음(동적 Typing) | 반드시 미리 정의 해야함 |
| 객체 참조 시기 | 실행 시 객체 참조 검사 (동적 바인딩) | 컴파일 시 객체 참조 검사 (정적 바인딩) |
| 보안 | Secure(HTML에 같이 수행되므로) But, 악의적인 목적에 코드 존재 | Secure  (Local File System에 Write 못하므로) |

**JavaScript 위치** – 외부형, <head>내부형, <body>내부형, 인라인형

\_js.html\_

<head>

<title>Page title</title

**<script><src=“myscript.js”></script>**

**<script>**

**Document.write(“head 내부형 자바스크립트<br>”);**

**</script>**

</head>

<body>

**<script>**

**Document.write(“body 내부형 자바스크립트<br>”);**

**</script>**

**<p onclick=“alert(‘인라인형 스크립트’)”></p>**

</body>

**\_myscript.js\_**

**Document.write(“외부 스크립트<br>”);**

**객체(Object)**

프로그래밍 대상을 속성과 동작으로 엮어서 정의한 자료형으로 속성(변수)와 메소드(함수)로 이루어진다. 🡪 **여러 속성을 하나의 변수에 저장할 수 있도록 해주는 데이터 타입**

객체 종류: 사용자 정의 객체(사용자가 정의한 객체), 내장 객체(JavaScript 엔진에 내장되어 있는 객체(String, Date, Math, Array, Number 등), BOM(브라우저를 제어할 수 있는 객체(window, screen, location 등)), DOM(Tree구조 HTML 요소에 접근할 수 있는 객체, HTML의 요소와 속성에 접근, 변경, 추가 그리고 삭제를 위한 표준 방법)

**쿠키(Cookie)와 세션(Session)**

**쿠키 (Cookie)**

**서버가 사용자의 웹 브라우저에 전송하는 작은 데이터 조각**을 의미한다.

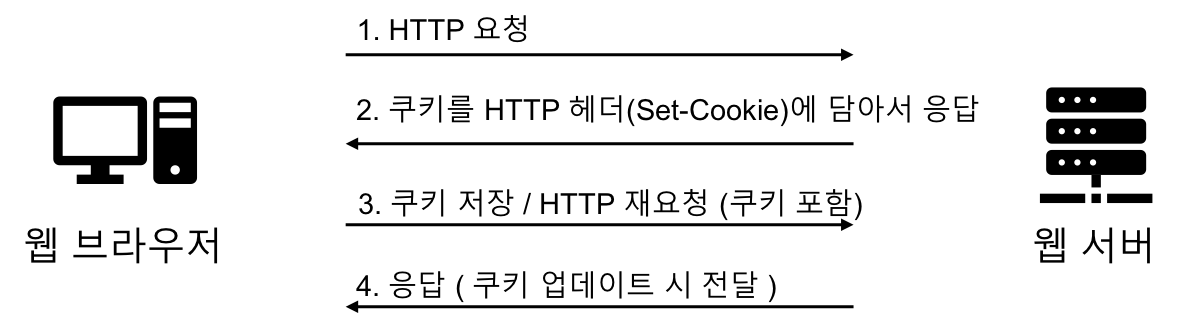
루 몬텔루라는 웹 브라우저 개발자가 웹 사이트에 접속한 클라이언트를 확인하기 **위해** 만들었다(stateless한 HTTP통신)

**쿠키는 주로 세션관리(서버에서 관리하는 로그인 등의 정보를 의미), 개인 설정유지, 사용자 트래킹(사용자의 행동을 기록하고 분석하는 것)용도로 사용된다.**

특징:

1. 한 개의 4KB까지 저장가능하며, 최대 300개까지 저장할 수 있는 텍스트 파일
2. **클라이언트에 저장됨**
3. **이름, 값, 만료날짜, 경로 정보가 들어있다.**
4. 기본적으로 쿠키는 웹 브라우저가 종료되면 삭제된다. (만료날짜 지정 시 만료일이 되야 삭제)
5. 웹 브라우저에 해당 서버의 쿠키 정보가 있으면 HTTP요청(HTTP헤더의 cookie)에 무조건 담아 보낸다.

작동 방식: HTTP통신에서 이루어지므로 HTTP의 응답과 요청에 따라 작동됨



**세션 (Session)**

통신을 하기 위해 **서로 연결된 순간부터 통신을 마칠 때 까지의 기간**을 의미

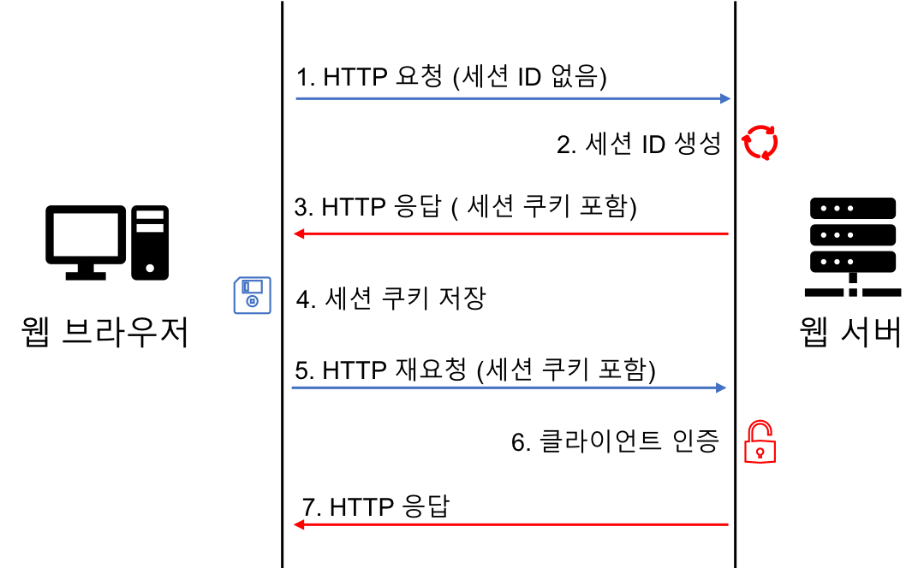
즉, HTTP세션이란 클라이언트가 웹서버에 연결된 순간부터 웹 브라우저를 닫아 서버와의 HTTP통신을 끝낼 때 까지의 기간이다.

**하지만 보통 세션은** 서버에 세션에 대한 정보(세션 상태, 클라이언트 상태, 세션 데이터 등)를 저장해 높고 클라이언트에게 세션 쿠키(고유한 세션 ID값)를 주어 **서버가 클라이언트를 식별할 수 있도록 하는 방식자체를 의미하는 경우가 많다.**

특징

1. 따로 용량의 제한이 없음 (서버의 능력에 따라 다를 수 있다)
2. 서버에 세션 객체를 생성하며 각 클라이언트마다 고유한 세션 ID값을 부여
3. 쿠키를 사용하여 세션 ID값을 클라이언트에게 보낸다.
4. 웹 브라우저가 종료되면 세션 쿠키는 삭제됨

작동방식



더 자세한 내용 🡪 <https://noahlogs.tistory.com/38?category=827412>

**JSP (Java Server Pages)**

**HTML내에 자바코드를 삽입**하여 **웹 서버에서 동적으로 웹 브라우저를 관리하는 언어**

* 보통 <% 자바코드 %> 식으로 사용

자바 언어를 기반으로 하는 **Server Side 스크립트 언어**

* 서버 측에서 동적인 컨텐츠 생성을 담당하는 기술
* 데이터베이스 연동 처리 등 다양한 구현이 가능
* 별도의 컴파일 과정 없이 HTML태그 수정 가능

**웹 애플리케이션?** 웹에서 실행되는 응용프로그램을 뜻하며 인터넷을 통한 은행업무, 인터넷 쇼핑 등 인터넷에서 하는 여러 서비스를 총칭 🡪 동작하기 위한 4가지 구성요소

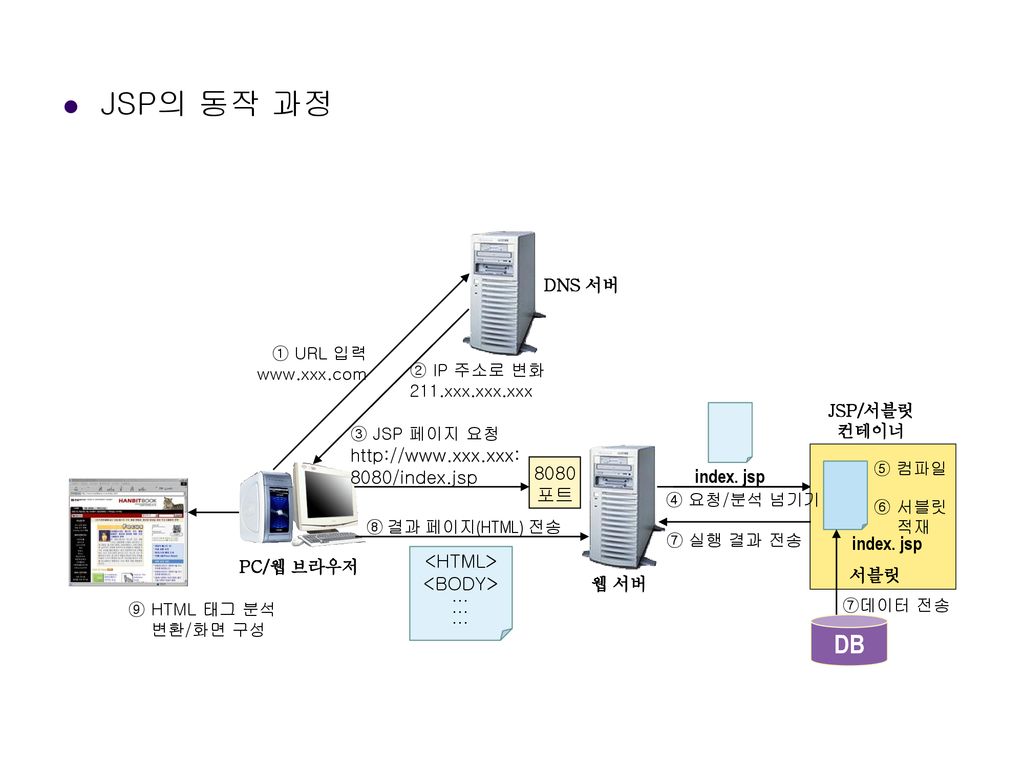
1. 웹 브라우저: 클라이언트가 요청하고 전달받은 페이지를 볼 수 있는 환경
2. 웹 서버: 클라이언트로부터 요청받아 서버에 저장된 리소스를 클라이언트에게 전달 (Apache)
3. 웹 애플리케이션 서버(WAS): 서버단에서 필요한 기능을 수행하고 그 결과를 웹서버에게 전달 (서블릿이나 JSP의 스크립트를 번역해줌, ex. Tomcat(톰캣))
4. 데이터베이스: 서비스에 필요한 데이터를 보관, 갱신 등 관리

**자바 서블릿(Java Servlet)?** 자바언어를 기반으로 웹페이지를 동적으로 생성하기 위한 서버측 프로그램이다. **웹 애플리케이션 서버위에서 컴파일되고 동작한다**. (**자바코드안에 HTML삽입 형태**)

**JSP로 작성된 프로그램은 서버로 요청 시 서블릿 파일로 변환되어 JSP태그를 분해하고 추출하여 다시 순수한 HTML로 변환하여 전송된다.**

<% %>형식의 약속된 기호들이 존재하고 이 기호들이 JSP가 서블릿으로 바뀔 때 어떻게 바꿀지 알려주는 부분이다. (자바코드)

**JSP동작과정**



특징

1. HTML코드안에 java코드가 있기 때문에 html코드를 작성하기 쉽다.
2. Servlet(수정하면 다시 컴파일해야함)과 다르게 jsp는 수정된 경우 재배포할 필요 없이 Tomcat이 알아서 처리해준다.
3. 자바 서블릿 API위에 만들어졌기 때문에 JDBC와 같은 자바 API들을 사용할 수 있다.
4. 자바 기반이므로 윈도우, 리눅스 운영체제 상관없이 플랫폼에 독립적이다.

**Request, response 객체**

JSP페이지를 제작하는 목적은 데이터 값을 전송하기 위함이다.

1. Request: 웹 브라우저(클라이언트)를 통해 서버에 어떤 정보를 요청하는 것으로 정보는 request객체가 관리한다. JSP컨테이너가 servlet으로 변환될 때 자동으로 생성된다. 주로 HTML폼을 통해 전달된 값을 가져올 때 사용
2. Response: 웹 브라우저의 요청에 응답하는 것으로 정보는 response객체가 관리한다. JSP컨테이너가 servlet으로 변환될 때 자동으로 생성된다. 사용자 요청에 대한 응답 처리, 페이지 전환

**포워딩과 리다이렉트**

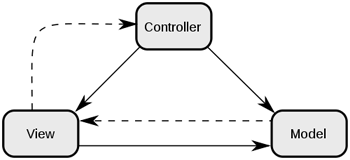
JSP환경에서 **페이지 전환 기능**으로 페이지 이동이 이루어지는 **방식차이가 존재**한다.

1. 포워딩(Forwarding): 웹 컨테이너 차원에서 페이지 이동만 하는 것으로 **실제로 클라이언트는 다른 페이지로 이동했는지 알 수 없다(주소 동일)**. 동일한 웹 컨테이너에 있는 페이지로만 이동할 수 있다. 클라이언트와 통신없이 **서버에서만 처리되는 것**으로 리다이렉트보다 나은 성능을 보인다. (request와 response객체를 공유) – **시스템(세션, DB, ..)에 변화가 생기지 않는 단순 조회 요청 경우 사용**
2. 리다이렉트(Redirect): sendRiderict()함수가 호출될 때 웹 컨테이너가 웹브라우저에게 다른 페이지로 이동하라고 명령한다. **클라이언트가 이동을 알 수 있다. (주소가 다름)** (새로운 request, response객체 생성) – **시스템 변화가 생기는 요청(로그인, 회원가입, 글쓰기 등)경우 사용**

**MVC 패턴**

어떠한 앱을 만들 때 유지보수를 하고 다른 이들과 공유를 하며 만들어야 할 때 좀 더 쉽고 깔끔하게 만들 수 있는 방법을 고안한 것을 디자인 패턴이라 하며 MVC패턴은 디자인 패턴 중 하나이다.

MVC는 Model, View, Controller의 약자로 하나의 애플리케이션, 프로젝트를 구성할 때 그 구성요소를 세가지의 역할로 구분한 패턴



Model: 애플리케이션의 **데이터(데이터베이스, 상수, 초기화 값, 변수 등)**를 나타내고 이러한 정보들의 가공을 책임지는 컴포넌트

View: 데이터 및 객체의 입력과 출력을 담당한다. 데이터를 기반으로 **사용자들이 볼 수 있는 화면**

Controller: 사용자가 데이터를 클릭하고 수정하는 것에 대한 “**이벤트”들을 처리하는 부분**으로 애플리케이션의 메인 로직을 담당한다. (model과 view의 중간다리역할)

장점: 유지보수성, 애플리케이션 확장성, 유연성(client의 새로운 요구사항에 대해 최소한의 비용으로 보다 유연하게 대처할 수 있는 것) 증가, 중복 코딩문제점 해결

**자바빈즈(JavaBeans)?** 컴포넌트 형태로 제작된 **자바 클래스 모듈**로 효율성, 재사용 등의 측면에서 장점

**JSP 빈즈?** 화면 로직 처리를 위한 JSP와의 연동을 위해 만들어진 자바 컴포넌트로 Beans액션을 통해 JSP에 손쉽게 연동 가능하다. **데이터베이스 연동이나 뷰 데이터 교환에 사용된다.**

JSP 빈즈 사용이유: 웹 애플리캐션의 OOP적 설계 가능. 컴포넌트 재사용 및 공유의 용이성. 프리젠테이션 로직과 비즈니스 로직의 분리 가능

**JSP와 JDBC**

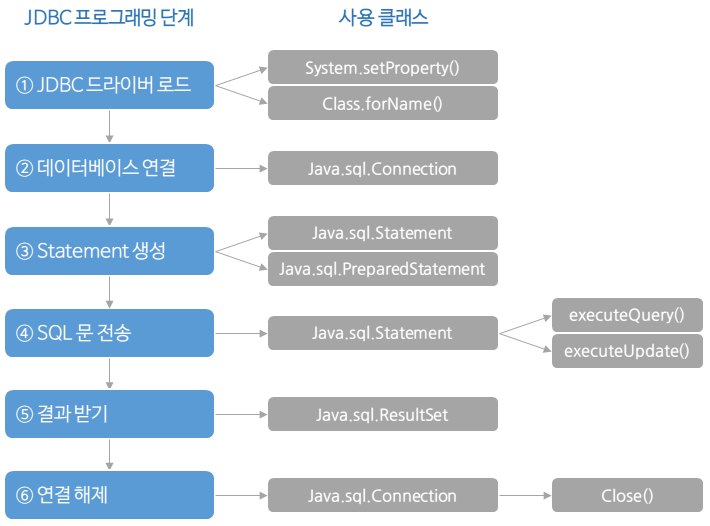
**JDBC and ODBC (Dynamic SQL) – open system**

**프로그램이 데이터베이스 서버와 interact하는 것을 돕는 API** | client와 server가 api를 통해 주고받기가 가능하다.

JDBC(JAVA Database connectivity): java에서 DB에 접속할 수 있도록 하는 API, 데이터베이스에서 자료를 쿼리하거나 업데이트하는 방법을 제공한다.

ODBC(Open database connectivity): 모든 응용 프로그램(프로그래밍 언어 무관)에서 모든 데이터베이스 시스템과 통신하는 데 사용할 수 있는 개방형 인터페이스. JDBC가 java만 지원하는 것과 다르게 ODBC는 언어에 대해 독립적이다.

데이터베이스 서버에 보낸 sql 명령에 대한 결과는 tuple 단위로 하나씩 가져와진다. (Fetch)



**트랜잭션?** DB에서 일련의 작업을 하나로 묶어 처리하는 것으로 JDBC는 commit과 rollback으로 구현 처리한다.

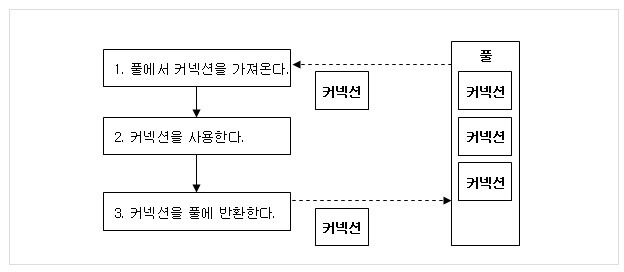
**커넥션?** 애플리케이션과 DB의 연결을 뜻하는 것으로, 애플리케이션에서 DB에 접속하고 종료하는 일련의 과정을 의미

동시 접속 사용자가 늘어날수록 프로그램에서 커넥션은 시스템에 많은 부하를 준다. 따라서 DB에는 일정 수준 이상의 커넥션이 연결되지 않도록 설정되어 있거나 라이선스 정책이 적용되어있다.

또한 사용하지 않는 커넥션으로 인한 DB 자원 낭비가 발생할 수 있다.

따라서 효율적인 커넥션 관리는 대규모 웹 애플리케이션 설계 시 성능과 안전성 면에서 중요한 요인이됨

**커넥션 풀?** 미리 일정한 수의 커넥션을 만들어 놓고 필요한 시점에 애플리케이션에 제공하는 서비스 및 관리 체계를 의미 (미리 생성하므로 DB에 부하를 줄이고 유동적 연결을 관리 가능)



1) 웹 애플리케이션 서버(웹 컨테이너)가 시작될 때 일정 수의 커넥션을 미리 생성해 풀을 구성한다.

2) 웹 애플리케이션 요청에 따라 생성된 커넥션 객체를 전달한다

3) 일정 수 이상의 커넥션이 사용되면 새로운 커넥션을 만든다.

4) 사용하지 않는 커넥션은 종료하고 최소한의 기본 커넥션을 유지한다.