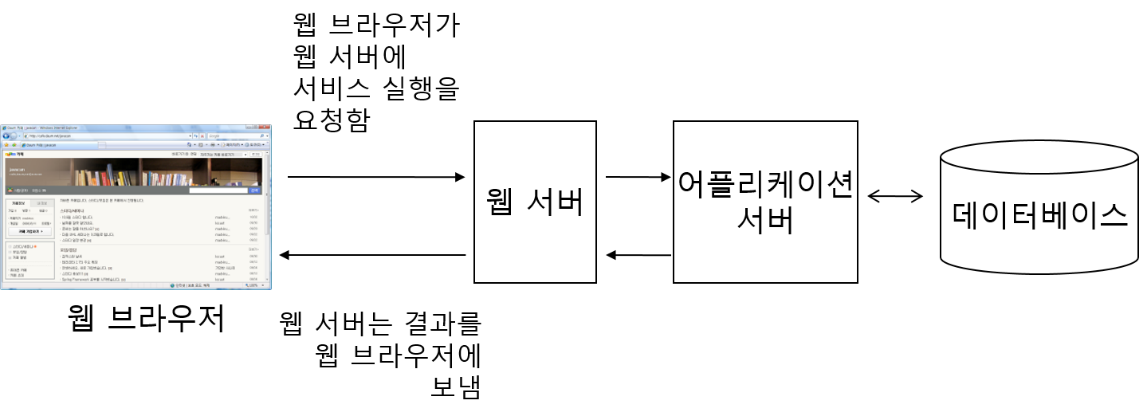
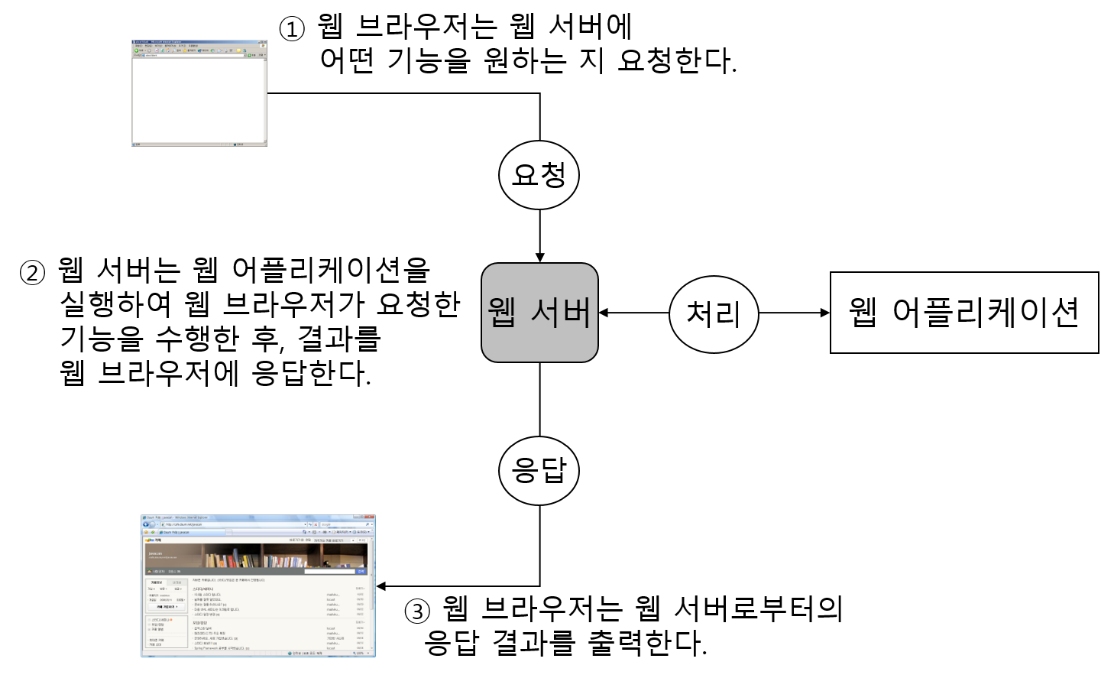
# 웹 프로그래밍 개요

## 웹 애플리케이션 구성요소



+) web server + application server = WAS

## 웹 애플리케이션 실행 순서

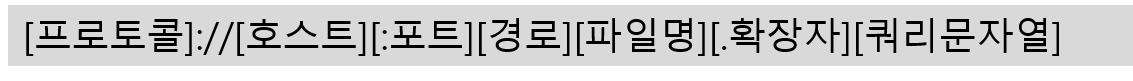


## 웹 컨테이너

웹 애플리케이션을 실행할 수 있는 컨테이너로 JSP와 서블릿을 실행.

주요 웹 컨테이너에는 아파치 톰캣(Apache Tomcat)이 있음.

## URL(Uniform Resource Locator)

* 구성 
  1. 예: http://www.google.com/search?hl=en&q=jsp&aq=f&oq=
     1. 프로토콜: http
     2. 호스트: www.google.com
     3. 포트: 80 (http 프로토콜의 기본 포트) – 기본 포트인 80은 생략이 가능하다.

또한, 미리 예약된 포트번호는 사용이 불가하며, 그 외의 포트번호는 임의 작성이 가능하다.

* + 1. 경로: /search
    2. 쿼리문자열: hl=en&q=jsp&aq=f&oq=
* URL은 웹 어플리케이션에 요청을 구분하기 위한 용도로 사용됨

# 서블릿과 JSP

자바를 만든 Sun에서 정한 웹 개발 표준

- 서블릿(Servlet) : 실행 코드 방식의 특징

- JSP(JavaServer Pages) : 스크립트 코드 방식의 특징

## JSP의 특징

자바 기반 스크립트 언어로 자바의 기능을 그대로 사용 가능.

HTTP에 대한 클라이언트의 요청 처리/응답.

웹 어플리케이션에서 결과 화면을 생성할 때 주로 사용.

-웹관련된 부분만 확인

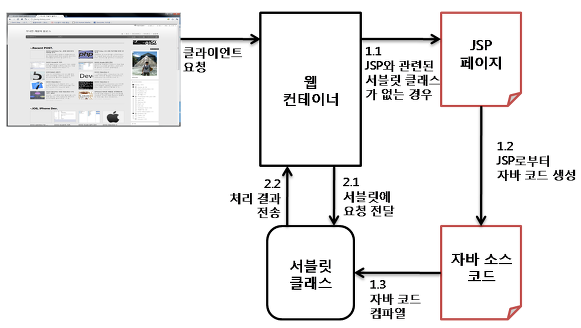
## JSP 사용 이유

자바 언어에 기반하기 때문에 플랫폼에 독립적(리눅스, 윈도우 등 운영체제에 상관없이 동작)

자바 언어에 대한 깊은 이해 없이도 초기 학습 가능(스크립트 언어는 상대적으로 자바 언어보다 단순)

스프링(Spring)이나 스트러츠(Struts)와 같은 프레임워크와 완벽하게 연동

## JSP 처리 과정



1. 웹 브라우저에 JSP 페이지에 해당하는 URL을 입력
2. JSP에 해당하는 서블릿이 존재하지 않을 경우(1.1)
   1. JSP 페이지로부터 자바 코드를 생성한다. (1.2)
   2. 자바 코드를 컴파일해서 서블릿 클래스를 생성한다. (1.3)
   3. 서블릿에 클라이언트 요청을 전달한다. (2.1)
   4. 서블릿이 요청을 처리한 결과를 응답으로 생성한다. (2.2)
   5. 응답을 웹 브라우저에 전송한다. (3)
3. JSP에 해당하는 서블릿이 존재하는 경우(1.1 ~ 1.3을 거친 경우)
   1. 서블릿에 클라이언트 요청을 전달한다. (2.1)
   2. 서블릿이 요청을 처리한 결과를 응답으로 생성한다. (2.2)
   3. 응답을 웹 브라우저에 전송한다. (3)

JSP 페이지를 요청할 때에는 JSP를 직접 실행하는 것이 아니라, JSP를 자바 소스 코드로 변환한 뒤 컴파일해서 생성한 서블릿을 실행

- JSP 페이지를 자바 코드로 변경하는 단계 -> “변환(translation)”

- 자바 코드를 서블릿 클래스로 변경하는 단계 -> “컴파일(compile)”

## 웹 애플리케이션 폴더 구성

|  |
| --- |
| * **WEB-INF**   : 웹 어플리케이션 설정 정보를 담고 있는 web.xml 파일이 위치한다.   * **WEB-INF\classes**   : 웹 어플리케이션에서 사용하는 클래스 파일이 위치한다.   * **WEB-INF\lib**   : 웹 어플리케이션에서 사용하는 jar 파일이 위치한다. |

### 웹 어플리케이션 폴더와 URL의 관계

한 웹 어플리케이션은 한 개의 폴더를 차지한다.

|  |
| --- |
| [톰캣]\webapps\[웹경로(프로젝트명)] -> http://host:port[\웹경로(프로젝트명)] |

### 웹 어플리케이션 폴더 내에서의 하위 폴더 사용

JSP 페이지를 기능별로 분류

## JSP 출력 과정

브라우저 출력을 위해 버퍼를 사용(출력버퍼와 브라우저 사이에 출력 스트림 존재)

