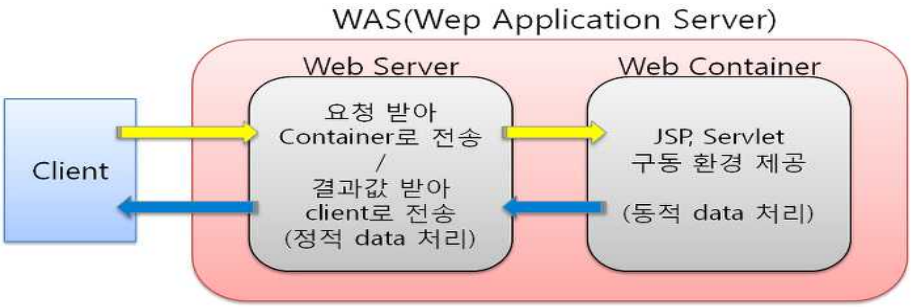
* Apache

: 아파치 소프트웨어 재단의 오픈소스 프로젝트이다.
: 웹서버로 불려진다
: 클라이언트 요청이 왔을 때만 응답하는 정적 웹페이지에 사용된다. (HTML, CSS, 이미지 등)
: 웹서버 - 80번 포트로 클라이언트 요청(POST, GET, DELETE)이 왔을 때만 응답

* Tomcat

: Servlet이나 JSP의 컨테이너
: WAS(Web Application Server)
: 컨테이너, 웹 컨테이너, 서블릿 컨테이너라고 부른다
: dynamic(동적)인 웹을 만들기 위한 웹 컨테이너



* 언어별 웹 서버 구성
(JAVA) JSP, Servlet -> 아파치 톰캣 -> WAS 서버 (WEB서버 내장)
(C, C++) PHP -> RWAPM -> WAS 서버 (WEB서버 내장)
(MFC, .NET) ASP, ASPX -> IIS -> WAS 서버 (WEB서버 내장)

1. 다운로드

tomcat.apache.org

apache-tomcat-10.1.18.exe

2. 설치

apache-tomcat-9.0.85.exe 더블클릭 → C:₩Tomcat 9.0 위치 선택

Tomcat이 설치되는 default 위치
C:₩Program Files₩Apache Software Foundation₩Tomcat 9.0

3. 서비스

① 시작 → 제어판 → 시스템 및 보안 → 관리도구 → 서비스
② 내PC → 우클릭 → 추가 옵션 표시 → 관리 → 서비스 및 응용 프로그램(왼쪽 창) → 서비스
③ 윈도우 + R → services.msc
시작유형 : 수동
서비스 : 중지

4. 서버실행

C:₩Tomcat 9.0₩bin₩Tomcat9.exe 더블클릭

5. Tomcat의 메인화면

http://localhost:8080

http://localhost:8080/index.jsp

http://127.0.0.1:8080

* 환경변수

JAVA_HOME (JDK의 위치)

C:₩Program Files₩Java₩jdk-17

TOMCAT_HOME or CATALINA_HOME (Tomcat의 위치)

C:₩Tomcat 9.0

[실습] hello.html

default Context : ROOT

webapps

http://localhost:8080/Context명/파일

http://localhost:8080/ROOT/hello.html - X ← default Context명 ROOT는 쓰시면 안된다.

http://localhost:8080/hello.html - O

ROOT가 아닌 다른 Context일 경우에는 반드시 URL에 써 주어야 한다.

Servlet

: **웹**에서 실행하는 프로그램
: html in JAVA
: public static void main(String[] args) 메소드가 없다
: 주기함수(Life Cycle)
: 반드시 public 이어야한다.
: new X
: 서버 안에 저장
: 반드시 어노테이션(@WebServlet) 또는 web.xml 등록해야 한다.

주기함수(Life Cycle)

init() : 맨 처음에 1번만 호출
↓
service() - doGet() : **클라이언트가 요청시마다 호출**
- doPost()
↓
destroy()

서비스

1. get방식
: **default**
: 주소표시줄(Query String)를 통해서 이동
: 이동되는 데이터가 보인다
: 이동되는 데이터가 **문자열만(String)** 처리

2. post방식
: **클라이언트가 post로 요청 시에만 적용**
: 내부적으로(페이지 단위) 이동
: 이동되는 데이터가 **안 보인다.**
: 대량 데이터

JSP (Java Server Page)

: 웹에서 실행하는 프로그램

: java in HTML

1. 선언문

<%! 전역변수 or 메소드 - 1번 처리 %> init()

2. 스크립트릿 (scriptlet)

<% 지역변수 or service처리 - 요청 시 마다 처리 %> service() - doGet(~~), doPost(~~)

3. 출력

<%= 값 or 변수 %>

[실습]

Context : testJSP

File : hello.jsp

http://localhost:8080/testJSP/hello.jsp

hello.jsp

↓

hello_jsp.java (서블릿)

↓ 소스가 바뀔 때만 컴파일

hello_jsp.class

내장객체

1. request : javax.servlet.http.HttpServletReqeuest
2. response : javax.servlet.http.HttpServletResponse
3. out : javax.servlet.jsp.JspWriter
4. session : javax.servlet.http.HttpSession
5. application : javax.servlet.ServletContent
6. pageContent : javax.servlet.jsp.PageContext
7. page : javax.servlet.jsp.HttpJspPage
8. config : javax.servlet.ServletConfig
9. exception : java.lang.Throwable

주석

1. Java

```
// 1줄
/*
    2줄 이상
*/
```

2. HTML

```
<!--
   웹브라우저에는 안보이나 소스보기(F12)하면 보인다.
   내부적으로는 처리( <%  %> <%=  %> 수행한다)
-->
```

3. JSP

```
<%--
   웹브라우저에도 안보이고 소스보기(F12)해도 안 보인다.
--%>
```

JSP directive (지시자)

JSP 페이지를 해당 서블릿으로 어떻게 변환하는지 웹 컨테이너에게 알려주는 메시지이다
응답에 대한 설정을 부여하는 것이다

- JSP page directive <%@ page %>
- JSP include directive <%@ include %>
- JSP taglib directive <%@ taglib %>

1. JSP page directive

현재 페이지에 대한 설정

JSP page directive 속성

- import
- contentType
  HTTP 응답의 MIME(Multipurpose Internet Mail Extension) 타입을 정의
  기본값(default)은 "text/html; charset=ISO-8859-1"
- extends
  extends 속성은 생성된 서블릿에서 상속할 상위클래스를 정의한다
  extends는 거의 사용되지 않는 속성이다
- info
  서블릿 인터페이스의 getServletInfo()에 의해 검색될 JSP 페이지의 정보를 설정한다
- buffer
  JSP 페이지에 의해 생성될 결과를 처리하기 위한 버퍼 사이즈를 KB단위로 설정한다.
  기본값(default)는 8KB 이다.
- language
- isELIgnored
  default=false이므로 EL이 사용되도록 설정되어 있다
  JSP에서 EL(Expression Language)를 무시하도록 한다
- isThreadSafe
  default=true
  JSP와 서블릿(servlet)은 모두 멀티쓰레드가 된다.
  속성을 false로 설정하면, 웹 컨테이너는 다수의 요청(request)을 순차적으로 실행할 것이다.
  즉, 하나의 요청에 대해 JSP가 응답한 후에 그 다음 요청을 하나씩 차례대로 수행하는 것이다
- autoFlush
  default=true
  출력할 내용이 굉장히 많아서 버퍼 용량을 넘을 경우, 자동으로 flush해서 자동으로 클라이언트에게 정보를 내보내게하는 속성이다
  버퍼가 기본값이면 autoFlush가 적용될 일이 없기 때문에 보통 false로 준다.
- session
  default=true
  로그인 정보 저장
- pageEncoding
  default=ISO-8859-1
- errorPage
  에러 페이지를 지정한다.

현재 페이지에 exception이 발생하면 설정된 에러 페이지를 호출한다.

- isErrorPage

  현재 페이지가 에러 페이지임을 선언한다.

2. JSP include directive

JSP, HTML, TEXT 등의 리소스 파일을 불러올 때 사용한다.
include는 페이지 번역(translation) 시 리소스 파일의 실제 내용을 불러온다.
똑같은 이름의 변수가 존재하면 error

3. JSP taglib directive

태그가 정의되어 있는 태그 라이브러리(tag library)를 지정할 때 사용한다.
태그를 정의하기 위해서는 TLD(Tag Library Descriptor)파일을 사용한다.

JSP Action Tag

JSP 페이지 내에서 자바 코드를 기술할 때 가독성을 높이기 위한 방법
JSP 페이지간 흐름 제어

1. useBean
   클래스 객체를 생성하고 사용할 범위를 지정한다
   <jsp:useBean id="" class="" scope="" />

   * scope 속성
   가. scope="page"
   - default
   - 현재 페이지에서만 적용
   나. scope="request"
   - request호출 관계에 있는 페이지간의 객체 공유
   - forward를 통한 페이지를 이동 했을 때
   다. scope="session"
   - 동일 브라우저(같은 세션)내에 표시되는 페이지간의 객체 공유
   라. scope="application"
   - 동일 서버내에서 서비스되어지는 페이지간의 객체 공유

2. setProperty
   useBean의 setter메소드 호출
   <jsp:setProperty name="" property="" value="" />

3. getProperty
   useBean의 getter메소드 호출
   <jsp:getProperty name="" property="" />

4. include

   다른 페이지를 현재 페이지에 포함한다
   <jsp:include page="" />
<jsp:include page="" /> 는 포함될 페이지를 먼저 실행하고 결과만 포함한다. 그래서 변수명이 똑같아도 error가 안난다
<%@ include file="" %> 는 먼저 포함하고 컴파일한다 그래서 변수명이 똑같으면 error가 난다

5. forward

   현재 페이지의 제어를 다른 페이지로 전달
   <jsp:forward page="" />

```
<%
    RequestDispatcher dispatcher = request.getRequestDispatcher(상대번지);
%>
```

Connectionless

HTTP 프로토콜은 클라이언트의 요청에 대한 응답을 하고 나면 해당 클라이언트와의 연결을 지속하지 않는다.

쿠키

: 생성된 쿠키는 클라이언트의 웹브라우저에 저장
: 웹사이트에 접속할 때 생성되는 정보를 담은 임시파일 (4KB)
: ID 기억 - 다음에 접속시 별도의 절차없이 빠르게 연결
: 쿠키 삭제는 시간을 0으로 셋팅
: 사생활 침해
: 팝업창의 오늘 하루 창 띄우지 않음
  새로고침으로 조회수 늘리는 걸 방지할 때
  최근 본 목록

쿠키생성

```
Cookie cookie = new Cookie("쿠키명", 값);
cookie.setMaxAge(3);//초 단위
```

세션

: 웹서버쪽의 웹컨테이너에 상태를 유지하기 위한 정보가 저장
: 세션은 기본 시간 1800초(30분)
: 각 클라이언트 고유 Session ID를 부여한다
  Session ID로 클라이언트를 구분하여 각 클라이언트 요구에 맞는 서비스 제공

세션 생성

```
HttpSession session = request.getSession();
session.setMaxInactiveInterval(30*60); //초 단위
```

세션 부여

```
session.setAttribute("세션명“, ”값“)
```

세션 얻어오기

```
session.getAttribute("세션명“)
```

세션 삭제

```
session.removeAttribute("세션명“)
```

모든 세션 삭제 - 무효화

```
session.invalidate()
```

Connection Pool

: 서버에 미리 Connection 를 설정해 놓는 것
: 데이터베이스와 연결된 커넥션을 미리 만들어서 풀(Pool)속에 저장해 두고 있다가 필요할 때 커넥션을 풀에서 가져다 쓰고 다시 풀에 반환(close)하는 기법
: Connection의 내용이 바뀌면 서버만 수정해주면 된다
: 풀속에 미리 커넥션이 생성 되어있기 때문에 커넥션을 생성하는데 드는 연결시간이 소비되지 않는다
: 커넥션을 계속해서 재사용하기 때문에 생성되는 커넥션 수는 많지 않다
: 오라클 주소, 드라이버, ID, PW를 서버에 숨겨 놓음으로 보안에 좋다
: 서버의 Connection 들을 얻어오려면 javax.sql.DataSource 를 이용
: server.xml에서 <Context></Context>에 추가해야하는데 따로 context.xml를 만들어서 사용해보자

: 필요한 라이브러리
commons-collections-3.2.1.jar
commons-dbcp-1.4.jar
commons-pool-1.6.jar

maxActive - 커넥션 풀이 제공할 최대 커넥션 개수
maxIdle - 사용되지 않고 풀에 저장될 수 있는 최대 커넥션의 개수, 음수일 경우 제한을 두지 않음
maxWait - 커넥션 풀을 대기하는 대기시간, 음수일 경우 제한을 두지 않음

EL (Expression Language), 표현언어

EL은 JSTL에 소개된 내용으로 JSP 2.0에 추가된 기능이며 JSP의 기본문법을 보완하는 역할을 한다

(1) EL에서 제공하는 기능

JSP의 네 가지 기본 객체가 제공하는 영역의 속성 사용
집합 객체에 대한 접근 방법 제공
수치 연산, 관계 연산, 논리 연산자 제공
자바 클래스 메소드 호출 기능 제공
표현 언어만의 기본 객체 제공
표기법 : ${ expr }

(2) 표현언어에서 자바메소드를 사용

- 자바클래스 작성하고 메소드는 static 설정
- 태그라이브러리에 대한 설정정보를 담고 있는 tld(Tag Library Descriptor)파일을 작성
- web.xml에 tld파일을 사용할 수 있는 설정정보를 추가
- 자바클래스에 접근하는 jsp파일을 작성

JSTL (Jsp Standard Tag Library)

2개의 jar파일이 필요

https://mvnrepository.com

jstl-1.2.jar

standard-1.1.2.jar

pageScope → requestScope → sessionScope → applicationScope 순으로 호출

page - pageScope
request - requestScope
session - sessionScope
application - applicationScope

메소드 호출시 접두사 set/get를 생략 할 수 있다
메소드명을 변수명처럼 사용 할 수 있다

제공되는 태그의 종류

라이브러리	URI	Prefix(접두어)
Core	http://java.sun.com/jsp/jstl/core	c
XML	http://java.sun.com/jsp/jstl/xml	x
국제화	http://java.sun.com/jsp/jstl/fmt	fmt
DB	http://java.sun.com/jsp/jstl/sql	sql
함수	http://java.sun.com/jsp/jstl/functions	fn

[실습] 회원가입과 로그인

[테이블]

```
create table member(
name varchar2(30) not null,
id varchar2(30) primary key, -- 기본키, unique, not null, 무결성 제약 조건
pwd varchar2(30) not null,
gender varchar2(3),
email1 varchar2(20),
email2 varchar2(20),
tel1 varchar2(10),
tel2 varchar2(10),
tel3 varchar2(10),
zipcode varchar2(10),
addr1 varchar2(100),
addr2 varchar2(100),
logtime date);
```

[실습] 방명록

[테이블]

```
create table guestbook(
seq    number primary key, -- 시퀀스 객체로부터 값을 얻어온다
name   varchar2(30),
email   varchar2(30),
homepage   varchar2(35),
subject   varchar2(500) not null,
content   varchar2(4000) not null,
logtime   date);
```

[시퀀스]

```
create sequence seq_guestbook nocycle nocache;
```

[실습] 게시판

[테이블]

```
CREATE TABLE board(
      seq NUMBER NOT NULL,                  -- 글번호 (시퀀스 객체 이용)
      id VARCHAR2(20) NOT NULL,             -- 아이디
      name VARCHAR2(40) NOT NULL,           -- 이름
      email VARCHAR2(40),                   -- 이메일
      subject VARCHAR2(255) NOT NULL,       -- 제목
      content VARCHAR2(4000) NOT NULL,      -- 내용

      ref NUMBER NOT NULL,                  -- 그룹번호
      lev NUMBER DEFAULT 0 NOT NULL,        -- 단계
      step NUMBER DEFAULT 0 NOT NULL,       -- 글순서
      pseq NUMBER DEFAULT 0 NOT NULL,       -- 원글번호
      reply NUMBER DEFAULT 0 NOT NULL,      -- 답변수

      hit NUMBER DEFAULT 0,                 -- 조회수
      logtime DATE DEFAULT SYSDATE
 );
```

[시퀀스]

```
CREATE SEQUENCE seq_board  NOCACHE NOCYCLE;
```

[실습] 이미지 게시판

[테이블]

```
CREATE TABLE imageboard(
      seq NUMBER PRIMARY KEY,
      imageId VARCHAR2(20) NOT NULL,       -- 상품코드
      imageName VARCHAR2(40) NOT NULL, -- 상품명
      imagePrice  NUMBER NOT NULL,        -- 단가
      imageQty     NUMBER NOT NULL,        -- 개수
      imageContent VARCHAR2(4000) NOT NULL,
      image1 varchar2(200),
      logtime DATE DEFAULT SYSDATE
 );
```

[시퀀스]

```
create sequence seq_imageboard  nocache nocycle;
```