Servlet

Java Web Application(Servlet, JSP)

1. **서블릿 생명주기**

Java web application은 진입점이 없기 때문에 LifeCycle에 의해 실행됨.

1-1 Init() : 서블릿 적재, 서블릿 초기화

1-2 service() : 웹브라우저 서비스 메소드

웹브라우저에 출력하는 함수 dopost와 doget으로 분기

1-3 destroy(): 서블릿 종료 직전 호출

서블릿과 JSP을 실행하기 위해서는 WAS(Web application server[tomcat,ignix]

1. **서블릿 생성 extends HttpServlet**

[HttpServlet extends genericservlet implements servlet]

Common Frame

public class mys extends HttpServlet{

init()

service(request,responde0

destroy()

}

Public void service(httpServletRequest req, HttpServletResponse resp){

Req: 요청객체 , Resp: 응답 객체

}

1. 기본 이론

GET url을 통하여 값을 전달 – 전달데이터가 한계가 있음.

Post는 Header에 담아 전달

서블릿은 웹어플리케이션에서 효율적인 자료공유방법을 제공한다.

콘텐츠와 비즈니스 로직을 분리할 수 있다.

컨트롤러와 뷰의 역할 분담으로 인해 웹디자이너와 개발자 간의 원할한 작업가능

유지보수 용이, 확장 용이

3 – 1 서블릿의 용도

1. UI로부터 Parameter 값 가져오기[request.getparameter]
2. 분석
3. DAO 모델객체 생성
4. Scope save [page context, request, session] - .setAttribute(“key”, value);

* JSP( getAttribute를 통하여 value 값 사용)
* Request는 Forward에서만 전달되나 Redirect는 Session에서 해야 값이 전달됨.

1. Page move = Forward, redirect

Web-INF 아래의 XML만 읽음

<servlet>

<servlet-name> my </servlet-name>

<servlet -class> com.ssafy.mysevlet </servlet-class>;

</servlet>

<Servlet-mapping>

<servlet-name> my</servlet-name>

<url –pattern> url /my.do</url-pattern>

</servlet-mapping>

한글깨질 때 encode 설정

setCharacterEncoding(“UTF-8”);

setcontentType(text/html;charset=UTF-8);

html parameter로 구분할 수 있도록 하는 value 두는 것도 좋은 방식

inline-block 특성과 div block의 특성 두 개를 다 가지는 블록으로 만듬.

bootstrap 최대 12개로 나눌 수 있음.

JSP (Java Server Page)

browser(HTML + CSS) + JSP[JAVA ENGIN]

영역 표시

**<%! %> 선언문**

**<% %> 처리문**

**<%= %> 출력문**

**<%-- --%> 주석**

**<%@ %> 메타데이터 및 설정 정보 [page, tag, include directive]**

**<%@page import>를 통하여 자바 라이브러리 및 클래스 가져오기 가능**

%{param.id} 방식으로도 가져올 수 있음.

MVC [MODEL VIEW CONTROL]

<MODEL2 구조>

장점: 업무 분담 용이, 유지보수 용이

1. Model

비즈니스로직과 관련된 부분 처리.

데이터베이스 관련 로직 구현

JSP Beans, EJB Component

애플리케이션의 데이터를 표현

1. VIEW

사용자에게 알맞은 화면을 보여주는 역할 수행

JSP, CUSTOMTAG

= 역할: 출력, 사용자 입력

1. Control

사용자에게 받은 데이터 처리

요청분석,

데이터 얻기 ,

모델 호출,

데이터저장 ,

뷰호출 = forward, redirect

MODEL1

jsp응답을 server 하나가 모두 해결하는 방식을 말함.

HTML + JAVA

MODEL2 -> Controller가 요청에 대해 제어를 하는 경우는 말함.

JSP+html view를 수행

Java : Controler and Model

Model의 범주 : business logic and Persistence layer

[Service, Dao, DTO]

DAO [Database access object]

db관련 업무담당

sql문을 실행하는 객체를 말함.

JSP 이동

RequestDispatcher patcher = request.getRequestDispatcher(“이동페이지”);

Dispatcher.forward(request,response);

Redirect – 완전 새 페이지

forward 차이점 – 이동은 하지만 현재 클래스 서블릿 데이터를 들고감. Request의 데이터를 다음 페이지로 전달함.

= url은 같으나 제공하는 정보가 다르다 그럼 forward다.

JSP의 4가지 영역

Page = 영역 별로 값입력

PageContext pageContext = new PageContext();

pageContext.setAttribute(“mykey”,동에번쩍);

Request 페이지 단위로 나누었을 때 영역

Session

HttpSession session = new HttpSession();

Session 브라우저를 닫지 않을 경우

Application

동일한 서버에 영역을 말한다. 같은 서버의 데이터를 저장하면 다른 컴퓨터도 저장된다.

JSP의 경로

ServerRoot 🡺 <http://localhost:8080>

ContextRoot 🡺 ./SSafy

Expression(표현식, 출력식)

주석 안에서 실행 시 HTML에서는 실행되지 않음.

EX)

<!-- <%= new Date %> -->

Javascript와 JSP는 서로 읽히지 않음.

단 다음과 같은 형태는 가능함.

단 String 형태의 변수의 경우 error가 발생함.

<script>

Document.write(<%=value%>); 다음과 같은 경우 먼저 JSP가 작동하고 이 후

Javascript가 작동해서 실행됨.

문자열의 경우

Document.write(‘<%=str%>’);

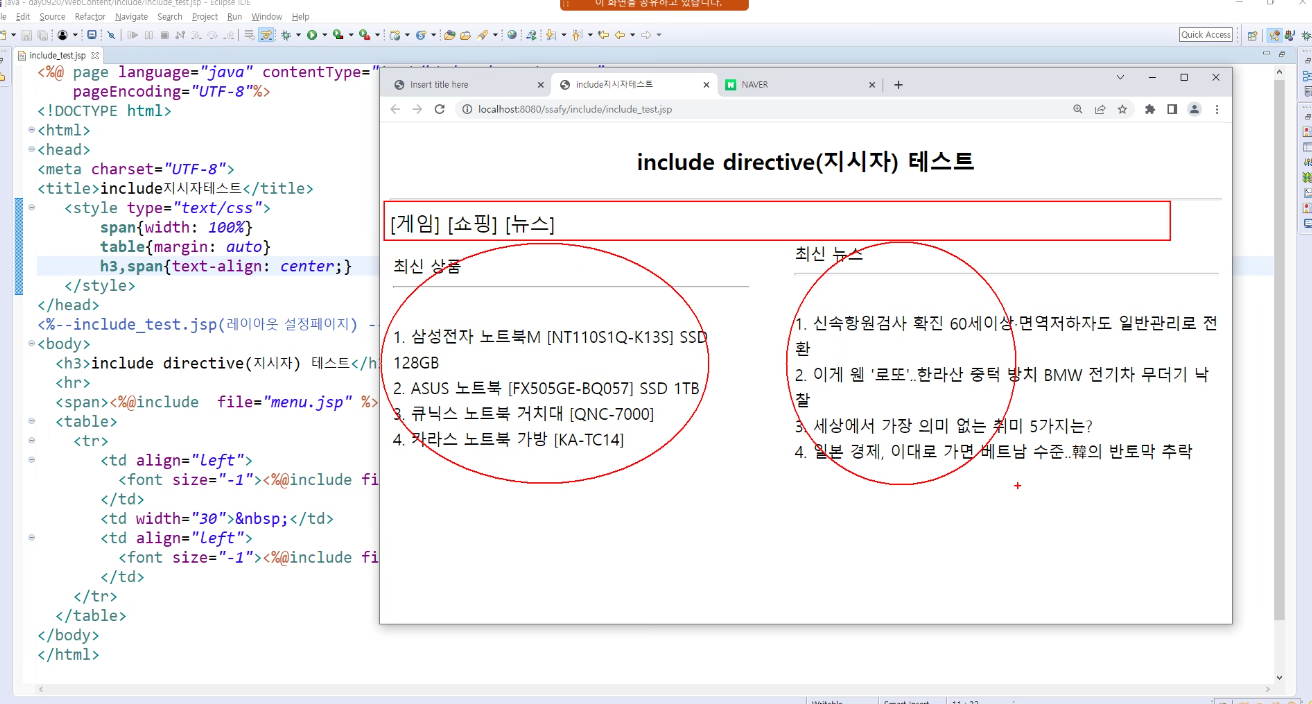
다음과 같이 값이 문자열임을 알려줘야함.

</script>

페이지 지시어

isErrorPage = Boolean 에러 발생시 에러 관련된 객체 지원

include: 여러 JSP를 하나의 JSP로 만들고 전체를 컴파일하는 방식



Include 한 파일에서 선언한 변수는 이 후 Include 페이지 내부에서도 사용이 가능함.

Menu.JSP = 변수 a = 30 선언

Shop.JSP에서 변수 <%= a%> 사용 시 출력이 됨.

Taglib 외부에서 정의된 태그 사용 시 사용하는 지시어

Set, Get Attribute는 request와 session에만 있다.

<input name = “id”>

<input name = “addr”>

getCookies()

getMethod()

getRemoteAddr()

getProtocol()

setCharacterEncoding() “post로 전달 시 인코딩”

예제

<%@page pageEncoding=”UTF-8” > 페이지에서 페이지로 넘어갈 때 Encoding을 맞춰줘야함.

Request 주요 메서드

getContextPath() = 현재 주소 가져오기.

Response 주요 메서드

setContentType() = 문자열 형태의 type에 지정된 MIME type으로 content type 설정

setHeader(name,value) = 문자열 name 이름으로 문자열 value 값 헤더로 설정.

setDateHeader(name,date) - name이름으로 date설정된 밀리세컨드 시간 값을 헤더에 설정

Url이 이전 값으로 바뀐 것이 아닌 경우 Redirect된거라 할 수 있음.

ServletContext Application

* Web Application
* Config – 한 개의 서블릿을 의미

Session

JSESSIONID -> value가 있는 지 확인 후 -> 없으면 서버에서는 번호 부여

주요 메소드)

Getid()

Getcreatingtime()

Invalidate() = 현재 세션 종료

setAttribute()

getAttribute()

removeattribute()

Cookie

서버에서 사용자 컴퓨터에 저장하는 정보파일

Key: value로 구성되나 String 형태로 이루어져있음.

JsessionID에 따라 저장될 수 있음.

-보안 취약- 클라이언트 사이드에 저장하기 때문에 변조가 가능함.

동작순서

Client가 페이지 요청

WAS Cookie 생성

Http Header에 Cookie 넣어 응답.

Browser 넘겨 받은 Cookie를 PC에 저장하고 다시 WAS 요청할 때 요청과 함께 Cookie 전송

동일 사이트 재방문 시 Client의 pc에 해당 Cookie가 있는 경우, 요청 페이지와 함께 전송

Cookie 특징

이름 값 만료일, 경로정보 구성

Client Maximum 300개 저장가능

쿠키 생성

Cookie coo = new Cookie(String name, String value);

쿠키 변경, 획득

Coo.setValue(key), coo.getValue(key)

사용 도메인지정/얻기

setDomain(domain), getDomain();

쿠기 유효기간 지정

Coo.setMaxAge(int expire);

Coo.getMaxAge();

쿠키를 client에 전송

Response.addCookie(cookie);

Client에 저장된 cookie 얻기

Cookie cookies[] = request.getCookies();

Class prefix 설명은: c++ namespace와 같음.

<java:class> <java:class>

<school:class> </school:class>

Expression language

Session 사용 시 사용

html부분에서 사용할 것

<% 내부에서 사용해선 안됨%>

문법 ${ 키값 또는 변수}

배열

${list[0]}

${list[1]}

${list[2]};

Custom 태그

JSP 페이지에서 반복 작업을 캡슐화하기 위하여 고안함.

1.2 방식의 커스텀태그

[%@taglib prefix=*"my"* uri=*"http://java.sun.com/jsp/jstl/core"* %](mailto:%25@taglib%20prefix=%22my%22%20uri=%22http://java.sun.com/jsp/jstl/core%22%20%25)

Prefix =”namespace”

<tag>

<name>태그내임 </name>

<

</tag>

JSTL

해당 태그가 참이 되지 않을 시 html태그처럼 무시함.

라이브러리 추가가 필요함.

Maven 사이트에서 JSTL dependency 복사 후

New -> Configure -> Convert Maven Project -> pom.xml에 붙여넣기할 것

작동이 안될 시에는 다운로드 받은 후 WEB-INF -> Lib 폴더 아래에 넣으면 추가가능.

<%@page import=*"java.util.ArrayList"*%>

<%@page import=*"java.util.List"*%>

<%@page import=*"com.ssafy.Person"*%>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<%@taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"* %>

<%@taglib uri=*"http://java.sun.com/jsp/jstl/core"* prefix=*"c"* %>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>JSTL테스트</title>

</head>

<%--jstl\_test.jsp --%>

<body>

<h3>JSTL(JSP Standard Tag Library)테스트</h3>

<hr>

<%-- <% if(조건식){

} %> --%>

<c:if test=*"true"* >

HTML마크업,text <br>

조건식 참일때 실행<br>

</c:if>

<c:if test=*"some text"*>

Some Text

</c:if>

<c:if test=*"*${2<3 }*"*>

2는 3보다 작다!! <br>

</c:if>

<%-- 나이테스트 --%>

<c:if test=*"*${param.age < 20}*"*>

<font color=*"blue"*>나이가 20미만입니다!!</font><br>

</c:if>

<c:if test=*"*${param.age >= 20}*"*>

<font color=*"blue"*>나이가 20이상입니다!!</font><br>

</c:if>

<c:if test=*"*${param.age >= 20 && param.age < 30}*"*>

<font color=*"blue"*>나이가 20대 입니다!!</font><br>

</c:if>

<hr>

<%-- 나이 20대,30대 체크 --%>

<c:choose> <%-- if ~ else if ~ else문과 유사 --%>

<c:when test=*"*${param.age<20 || param.age>39}*"*>

<font color=*"green"*>나이가 20대 또는 30대가 아닙니다!!</font><br>

</c:when>

<c:when test=*"*${param.age < 30 }*"*> <%-- 20~39 --%>

<font color=*"green"*>나이가 20대 입니다!!</font><br>

</c:when>

<c:otherwise>

<font color=*"green"*>나이가 30대 입니다!!</font><br>

</c:otherwise>

</c:choose>

<hr>

<h4>forEach테스트</h4>

<%-- JSTL안녕!! 홀수줄은 빨강, 짝수줄은 파랑으로 총 10줄을 출력하시오 --%>

<c:forEach begin=*"0"* end=*"10"* var=*"i"* varStatus=*"stat"* step=*"2"*>

인덱스: ${stat.index}, 카운트: ${stat.count}

JSTL안녕!!==>${i }<br>

</c:forEach>

<%--

var="i" ==> 새로 정의하는 변수 (begin,end에 정의된 숫자를 사용하기 위해)

int i=1~10

varStatus = "gildong" ==> 출력되는 각 행에 번호를 부여 하기 위해

gildong.index gildong.count

정의된 값 1부터 시작하는 값

--%>

<c:forEach begin=*"1"* end=*"10"* var=*"i"* varStatus=*"stat"*>

<c:if test=*"*${stat.count %2 == 0 }*"*>

<font color=*"blue"*>JSTL안녕!!</font><br>

</c:if>

<c:if test=*"*${stat.count %2 == 1 }*"*>

<font color=*"red"*>JSTL안녕!!</font><br>

</c:if>

</c:forEach>

<%

//서블릿(Controller) ==> 영역에 데이터를 저장

Person p1 = **new** Person("길동",13,"학생");

Person p2 = **new** Person("라임",14,"학생");

Person p3 = **new** Person("주원",15,"학생");

List<Person> list = **new** ArrayList<>();

/\* list.add(p1);

list.add(p2);

list.add(p3); \*/

request.setAttribute("list", list);

%>

<hr>

<h3>Person정보</h3>

<!-- 테이블 출력 -->

<%-- <c:if test="${list != null}">--%>

<%-- <c:if test="${list ne null}"> --%>

<%-- <c:if test="${! empty list}"> --%>

<c:if test=*"*${**not empty** list}*"*>

<table>

<tr bgcolor=*"skyblue"*>

<th>이름</th>

<th>나이</th>

<th>직업</th>

</tr>

<c:forEach items=*"*${list}*"* var=*"p"*>

<tr>

<td>${p.name }</td>

<td>${p.age }</td>

<td>${p.job }</td>

</tr>

</c:forEach>

</table>

</c:if>

<c:if test=*"*${**empty** list}*"*>

정보가 없습니다.

</c:if>

<%--

List<Person> list = request.getAttribute("list");

<table>

for(){

Person p= list.get(i);

String name = p.getName();

int age = p.getAge();

String job = p.getJob();

<tr>

"<td>"+name+"</td>"

"<td>"+age+"</td>"

"<td>"+job+"</td>"

</tr>

}

</table>

--%>

</body>

</html>

Jstl 예시 코드

DAO Database access object

Database 접근 및 수행 업무 수행

Ex) select, update, remove, selectall etc….

Service 객체

DAO을 singleton 방식으로 가지고 있음.

심화에서는 ThreadPool객체를 가지고 있을 것임.

Controller에서 실행할 기능을 탑재한 객체 singleton 방식으로 보통 구현되서 가지고 있음.

Service object

user

Content

DAO sample code

Base code

Connection conn = null;

PreparedStatement pt = null;

Try{

Conn = getConnection();

Pt = conn.preparedStatement(“sql”);

조회 함수{

ExcuteQuery return ResultSet or null; // select

ExcuteUpdate return int; // insert, update, delete

}

}

Select{

String sql =”select \* from tablename where number = ?;

}

Insert{

Sql = “insert into table(column1,column2 …. ) value(?,?,?);

}

Update{

String sql = “update table set val=? Val2=? Val3=? Where number = ?”;

}

Finally{

Conn.close(); // close는 역순으로 resultset -> preparedstatement -> connection

Pt.close();

If(select) Resultset.close();

}

Db connection Code{

package com.ssafy.sample.util;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DBUtil {

private DBUtil() {

try {

Class.forName("com.mysql.cj.jdbc.Driver");

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

}

private static DBUtil instance = new DBUtil();

public static DBUtil getInstance() {

return instance;

}

public Connection getConnection() throws SQLException {

String url =

"jdbc:mysql://127.0.0.1:3306/ssafytest?serverTimezone=UTC&useUniCode=yes&characterEncoding=UTF-8";

String user = "ssafy";

String pwd = "ssafy";

return DriverManager.getConnection(url, user, pwd);

}

public void close(AutoCloseable... closeables) {

for (AutoCloseable c : closeables) {

if (c != null) {

try {

c.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

}

Resultset 데이터 가져오기

GetInt(1) int형으로 가져오기

Getstinrg(2) string 형으로 가져오기

Getstrnig(column name)

Backend Progress

<작업 순서>

1. Create

inputForm

-> Controller get -> service -> DAO(sql insert)

2. Read(목록, 상세)

Controller를 통한 조회 -> Service -> DAO(sql select)

-> 컨트롤러 저장(영역 request/session) = setAttribute

-> 출력 jsp 이동 = getAttribute

3. Update(수정)

Read를 통한 데이터 조회 -> service -> DAO

Controller 저장 -> setAttribute

수정폼.jsp -> getAttribute

==> 수정폼 JSP 기존데이터 수정

==> 컨트롤러 GET -> Service - DAO (sql update)

4. Delete

==> 컨트롤러 GET -> Service - DAO (sql delete)

=======================

5. Login/ logout

==> login data select

==> 세션처리 httpsession session = request.getsession()

==> session.setattribute("loginuser", data);

logout

getsession -> session.invalidate;

redirect

추가적으로 new Cookie addCookie

6. paging

==> 3중 쿼리 : limit (top N)