Week5 Review

* 웹서버: 웹에서 서버 기능을 수행하는 프로그램으로서 HTTP라는 프로토콜을 기반으로 하여 웹 클라이언트(브라우저)로부터의 요청을 서비스하는 기능을 담당
* 웹 애플리케이션 서버(WAS): 크게 웹 서버 기능와 컨테이너 기능으로 구성. 톰캣은 일반 상용 WAS처럼 JavaEE 스펙을 모두 갖추고 있지 않고 JSP와 서블릿을 실행하는 컨테이너와 웹서버만 제공
* 서블릿(Servlet):
* 자바를 이용하여 웹에서 실행되는 프로그램을 작성(자바 코드를 html로 변환)하는 기술.
* 클라이언트 요청을 처리하고 그 결과를 다시 클라이언트에게 전송하는 Servlet 클래스의 구현 규칙을 지킨 자바 프로그램
* 자바 클래스 형태의 웹 애플리케이션. 브라우저를 통해 자바 클래스가 실행되도록 하기 위해서 javax.servlet.http 패키지에서 제공하는 HttpServlet 클래스를 상속받아 구현해야 한다. HttpServlet 클래스를 상속받아 만든 서브클래스를 서블릿 클래스라고 함
* 서블릿컨테이너: 서블릿의 생명 주기를 관리하고 JSP를 서블릿으로 변환하는 기능을 수행하며 요청에 따른 스레드를 생성, 클라이언트의 Request 를 받아주고 Response 를 보낼 수 있게 웹 서버와 소켓을 만들어서 통신을 해 줌.
* JSP(Java Server Page):
* 서블릿은 문법이 어려운 데 (복잡한 메소드를 구현, 코드를 일일이 작성..), HTML 소스 안에 <%’, ‘%> 문자(JSP태그)만 쓰고 바로 자바 코드 쓰면 됨
* 개발자가 jps를 만들어서 실행하면 톰캣이 서블릿으로 변환시켜줌: jsp -> 서블릿 -> html
* 지시자태그: 페이지의 속성을 나타내거나 다른 jsp파일 include할 때 사용 <%@ %>

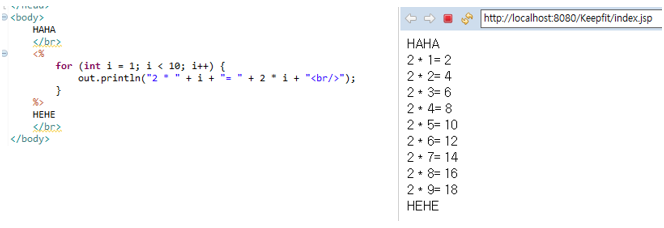
-Page: <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

-Incude: 별도의 페이지를 현재 페이지에 삽입

-taglib : JSTL 할 때 사용.. 사용자가 만든 태그들 (태그 라이브러리)을 사용

* 주석태그 <%-- --%>
* 선언태그: 변수나 메소드 선언 시 사용 <%! %>
* 표현식: JSP에서 **결과 값을 바로 출력**해줄 때 사용 <%= %>
* 스크립트릿: 일반 자바 코드를 사용할 수 있는 태그 <% %> (옛날식)



* 액션태그: 자바 빈을 연걸. JSP페이지 내에서 어떤 동작을 하도록 지시하는 태그. 표준 액션 태그는 특별한 선언 없이 "jsp"라는 접두어를 붙여 태그명만 명시하면 컨테이너가 인식해서 수행

- **forward**: 다른 페이지로 이동할 때 <jsp:forward page="경로" /> or <jsp:forward page="<%= %>" /> \*url 주소는 변경되지 않고 페이지 내용만 바뀜

- **include:** 다른 페이지를 현재 페이지에 포함하고자 할 때 <jsp:include page="경로" /> or <jsp:include page="<%= %>" />

- **useBean:** 새로운 객체를 생성 <jsp:useBean class="HelloBean" id="hello">

- **param:** forward/include 태그에 데이터 전달. 이름 & 값으로 구성

<jsp:forward page=“include02.jsp”>

<jsp:param name=“id” value=“abc”/>

<jsp:param name=“pw” value=“123”/>

</jsp:forward>

\* request.getParameter("name") 으로 받음

**- JSTL:**

위에서 본 JSP 액션태그, 스크립트릿은 mvc모델을 지키지 않음. 뷰단에서만 사용하는 JSTL은 jsp에서 태그만으로 구성하고 자바 문법을 사용하지 말자는 취지 -> 자바 컴파일 과정을 거치지 않으므로 서버 재기동할 필요가 없음

* Html로 지원하는 tag를 확장해서 jsp에서 개발자가 활용하는 태그를 만든 것을 커스텀 태그라고 함. 커스텀 태그 중에서 많이 사용되는 것을 모아서 jstl이라는 규약을 만듬.
* 태그 종류:

1) 코어 (<c:XXX>): 변수, 흐름 제어, url 처리

2) 국제화(<fmt: XXX>): 숫자 및 날짜 형식 처리

3) XML(prefix: x): xml 문서를 처리할 때 필요한 기능 제공

4) DataBase(prefix: sql): DB의 데이터를 입력, 수정, 삭제 조회

5) Function(prefix: fn): 문자열을 제공하는 함수

* 활용: 라이브러리 다운받아 WEB-INF/lib 위치 – jsp 상단에 <%@ taglib prefix=”c” uri=<http://java.sun.com/jsp/jstl/core> %>
* Core Library

1. 페이지 합치기

[welcome.jsp]

안녕하세요! <br>

<br>

<jsp:include page=*"test1.jsp"*></jsp:include>

[test1.jsp]

로그인 정보를 입력해주세요.

<form action=*"login"* method=*"post"*> //이 value 값은 controller의 model.addAttribute에서 옴

<div><input type=*text* name=*"id"* value=*"*${id}*"* placeholder=*"아이디"* ></div>

<div><input type=*password* name=*"pw"* value =*"*${pw}*"* placeholder=*"비밀번호"*></div>

<div><input type=*text* name=*"email"* value=*"*${email}*"* placeholder=*"이메일"*></div>

<div><input type=*text* name=*"phone"* value=*"*${phone}*"* placeholder=*"폰"*></div>

<button type=*"submit"*>로그인</button>

</form>

1. 사용자 정보 맵으로 만들어서 출력하기

@RequestMapping("/login") //model: key-value 쌍을 view단으로 보냄

**public** String method3(String id, String pw, String email, String phone, Model model){

**if** ( "lia".equals(id) && "1234".equals(pw)) {

Map userinfo = **new** HashMap();

userinfo.put("id", id);

userinfo.put("email", email);

userinfo.put("phone", phone);

model.addAttribute("userinfo", userinfo);

**return** "test/login";

}

**else** **if** ("".equals(id)) {

**return** "test/loginFail";

} **else** {

**return** "test/loginFail";

}

}

[login.jsp]

안녕하세요!! :) <br>

고객님의 정보를 확인해주세요. <br>

아이디: ${userinfo.id}<br>

이메일: ${userinfo.email}<br>

폰: ${userinfo.phone}<br>--%>

1. 사용자 정보 리스트로 만들어서 출력하기

@RequestMapping("/login")

**public** String method3(String id, String pw, String email, String phone, Model model){

**if** ( "lia".equals(id) && "1234".equals(pw)) {

List userinfo = **new** ArrayList();

userinfo.add(id);

userinfo.add(email);

userinfo.add(phone);

model.addAttribute("userinfo", userinfo);

**return** "test/login";

}

**else** **if** ("".equals(id)) {

**return** "test/loginFail";

} **else** {

**return** "test/loginFail";

}

}

[login.jsp]

안녕하세요!! :) <br>

고객님의 정보를 확인해주세요. <br>

<c:forEach var =*"info"* items=*"*${userinfo}*"*> <%--변수는 직관적인 이름 --%>

${info} <br>

</c:forEach>

1. 테이블로 출력하기

안녕하세요!! :) <br>

고객님의 정보를 확인해주세요. <br>

<table>

<table border = *1*>

<tr>

<th>아이디</th> <th>이메일</th> <th>폰</th>

</tr>

<tbody>

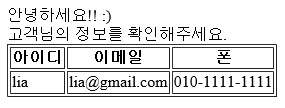
<c:forEach var =*"info"* items=*"*${userinfo}*"*>

<td>${info} </td>

</c:forEach>

</tbody>

</table>



* **리눅스**
* sh파일 만들어보기 $vim clear.sh
* 생성된 sh 파일 실행하기 (동일한 경로상의 화면에서 실행) $./clear.sh
* 파일 실행가능하도록 변경하기 $chmod +x clear.sh
* 반복 예약작업 **cron**

등록형식 \* \* \* \* \* (분/시/일/월/요일) 수행할 명령어

\*\*\*\*\*/root/every\_1min.sh -> 1분마다 수행

0 8 \* \* 1—5 /root/weekday.sh -> 평일 8시되면 수행

1. 파일 입력받아서 정보 추출한후 테이블에 넣어서 화면에 출력

[Controller.java]

@Controller

**public** **class** TestController401 {

@RequestMapping("/main")

**public** String method1(Model model) {

File file = **new** File("C:\\Users\\이정화\\Documents\\workspace-sts-3.9.8.RELEASE\\pjcf2\_sample\_web401\\src\\main\\resources\\pjcf\\crypto.txt");

**try** {

FileReader filereader = **new** FileReader(file);

BufferedReader bufReader = **new** BufferedReader(filereader);

String line ="";

List<HashMap<String, Object>> contents = **new** ArrayList();

**while** ((line = bufReader.readLine()) != **null**) {

String[] content = line.split("\\|");

Map <String, Object> map = **new** HashMap();

map.put("name", content[0]);

map.put("link", content[1]);

map.put("date", content[2]);

map.put("gubun",content[3]);

contents.add((HashMap) map);

}

model.addAttribute("contents", contents);

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** "test/welcome";

}

[welcome.jsp]

안녕하세요! <br>

<br>

<table border =*1*>

<tr>

<th>서버</th> <th>날짜</th>

</tr>

<tbody>

<c:forEach var=*"info"* items=*"*${contents}*"*>

<c:if test=*"*${info.gubun != -1}*"*> // “{info.gubun}”

<tr>

<td><a href=*"*${info.link}*"*>${info.name}</a></td>

<td>${info.date}</td>

</tr>

</c:if>

</c:forEach>

</tbody>

</table>

1. 로그인 정보 입력받은 후 그 정보를 파일에 출력

@Controller

**public** **class** TestController402 {

@RequestMapping("/main2")

**public** String method(Model model) {

**return** "test/test1";

}

@RequestMapping("/login2")

**public** String method(String id, String pw, Model model) **throws** IOException {

System.***out***.println(id + ": " + pw);

@SuppressWarnings("resource")

BufferedWriter oos = **new** BufferedWriter(**new** FileWriter("C:\\Users\\이정화\\Desktop\\accountInfo.txt"));

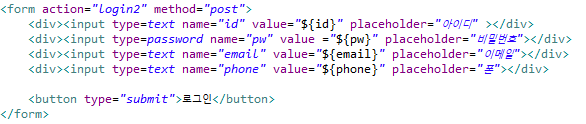
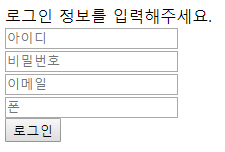
oos.write(id + ": " + pw);

oos.flush();

**return** "test/login2";

}

}



1. 파일에서 게시글 목록을 받아와 화면에 보여주고, 게시글 클릭시 해당 내용 보여준 후 조회수 업데이트해서 파일에 출력

@Controller

**public** **class** TestController403 {

@Resource

**private** Properties configProp;

//E:/data\_project401/cryptoList.txt

@RequestMapping("/main3")

**public** String method(Model model) {

String filePath = configProp.getProperty("filePath");

System.***out***.println(">>>>"+filePath);

//E:/data\_project401/

File file = **new** File(filePath+"cryptoList.txt");

**try** {

FileReader filereader = **new** FileReader(file);

BufferedReader bufReader = **new** BufferedReader((filereader));

String line ="";

List<HashMap<String, Object>> contents = **new** ArrayList();

**while** ((line = bufReader.readLine()) != **null**) {

String[] content = line.split("\\|" , -1);

Map <String, Object> map = **new** HashMap();

map.put("no", content[0]);

map.put("name", content[1]);

map.put("link", content[2]);

map.put("date", content[3]);

map.put("view",content[4]);

map.put("gubun",content[5]);

contents.add((HashMap) map);

System.***out***.println(contents.toString());

}

model.addAttribute("contents", contents);

} **catch** (IOException e) {

e.printStackTrace();

}

**return** "test/board";

}

@RequestMapping("/test/content")

**public** String method2 (String page, Model model) {

String filePath = configProp.getProperty("filePath");

System.***out***.println(page);

File file2 = **new** File(filePath+"cryptoList.txt");

List<String> lines = **new** ArrayList();

String line ="";

**try** {

FileReader filereader2 = **new** FileReader(file2);

BufferedReader bufReader2 = **new** BufferedReader((filereader2));

**while** ((line = bufReader2.readLine()) != **null**) {

String[] content = line.split("\\|", -1);

**if** (page.equals(content[0])) {

line = line.replace(content[4], Integer.*toString*((Integer.*parseInt*(content[4]) + 1)));

lines.add(line);

}

**else** {

lines.add(line);

}

}

filereader2.close();

bufReader2.close();

//

// System.out.println(lines.toString());

FileWriter fos = **new** FileWriter(filePath+"cryptoList.txt");

BufferedWriter bw = **new** BufferedWriter(fos);

**for**(String line2: lines) {

bw.write(line2 + "\r");

}

bw.flush();

bw.close();

File vfile = **new** File(filePath+"cryptoView.txt");

FileReader vfilereader = **new** FileReader(vfile);

BufferedReader vbufReader = **new** BufferedReader(vfilereader);

String vline ="";

List<HashMap<String, Object>> vcontents = **new** ArrayList();

**while** ((vline = vbufReader.readLine()) != **null**) {

String[] content = vline.split("\\|", -1);

**for** (String no : content) {

**if**(page.equals(no)) {

Map <String, Object> map = **new** HashMap();

map.put("no", content[0]);

map.put("con", content[1]);

map.put("writer",content[2]);

model.addAttribute("map", map);

}

}

}

} **catch** (Exception e) {

e.printStackTrace();

}

**return** "test/content";

}

}