77일차

코딩

2024/04/18 17:33 http://blog.naver.com/cy_109/223419666631

♪©☎☎Ⓚ오늘 개인 프로젝트 목표 화면 구현 다하기◆££→

76일차

java

2024/04/17 13:39 http://blog.naver.com/cy_109/223418243502

SMTP

메일 인증 시스템 만들기

〈네이버 메일 서버를 이용한 메일 인증〉

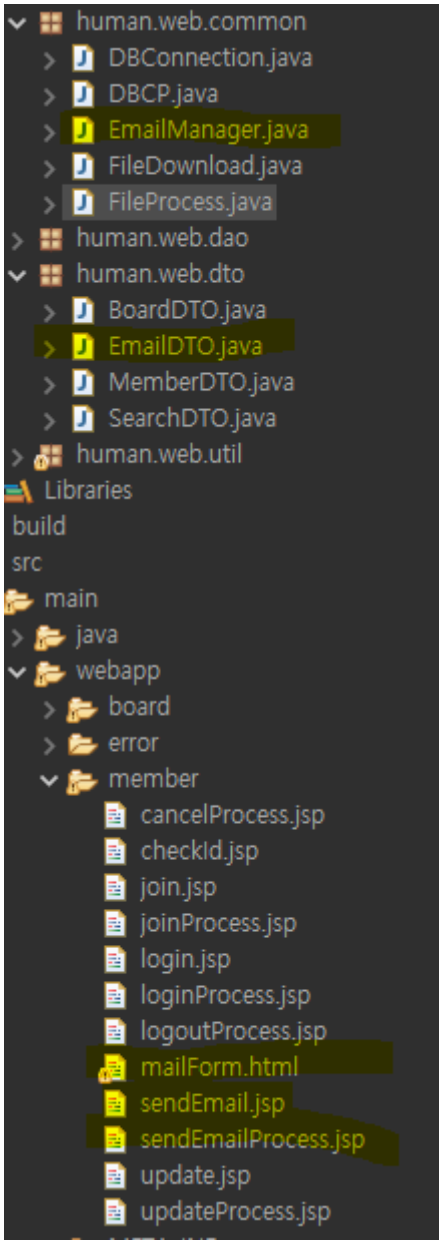
1. 네이버 로그인 > 메일 > 환경설정 > POP3/IMAP 설정에서 선택
-POP/SMTP 적용:사용함
-적용범위:지금부터 새로받은 메일만 받음 / 이전에 설정한 시간 이후 수신한 메일만 받음
-읽음표시: POP3로 읽어간 메일을 읽지 않음으로 표시
-원본저장: 메일 프로그램 설정에 따라 저장 또는 삭제
-외부메일 처리:POP3로 읽어갈 때 외부메일을 포함하지 않음

smtp 서버명: smtp.naver.com
smtp 포트: 465, 보안 연결(SSL) 필요

2. 메일 보내는 페이지 작성: sendEmail.jsp

3. 메일정보를 저장할 수 있는 DTO 작성: EmailDTO.java

4.메일관련 처리를 할 수 있는 Java 클래스 작성: EmailManager.java
- 필요한 메일 관련 API 다운로드 및 lib 폴더에 추가
- mvnrepository.com
- JavaMail API (compat) >> 1.4.7 : mail-1.4.7.jar 다운로드
- JavaBeans(TM) Activation Framework >> 1.1.1 : activation -1.1.1.jar 다운로드



MailDTO.java

package human.web.dto; public class EmailDTO { private String from;//보내는 사람의 메일 주소 private String to;//받는 사람의 메일 주소 private String subject;//제목 private String content;//내용 private String format;//보내는 내용의 형식(MIME 타입) public String getFrom() { return from; } public void setFrom(String from) { this.from = from; } public String getTo() { return to; } public void setTo(String to) { this.to = to; } public String getSubject() { return subject; } public void setSubject(String subject) { this.subject = subject; } public String getContent() { return content; } public void setContent(String content) { this.content = content; } public String getFormat() { return format; } public void setFormat(String format) { this.format = format; } }

MailManager.java

package human.web.common; import java.util.Properties; import javax.mail.Authenticator; import javax.mail.Message; import javax.mail.MessagingException; import javax.mail.PasswordAuthentication; import javax.mail.Session; import javax.mail.Transport; import javax.mail.internet.AddressException; import

javax.mail.internet.InternetAddress; import javax.mail.internet.MimeMessage; import human.web.dto.EmailDTO; public class EmailManager { private final Properties serverInfo;//서버정보 private final Authenticator auth;//인증정보 //인스턴스 블록을 이용한 인스턴스 필드 초기화 { serverInfo = new Properties(); //네이버 SMTP 서버 접속정보 serverInfo.put("mail.smtp.host", "smtp.naver.com"); serverInfo.put("mail.smtp.port", "465"); serverInfo.put("mail.smtp.socketFactory.port", "465"); serverInfo.put("mail.smtp.starttls.enable", "true"); serverInfo.put("mail.smtp.auth", "true"); serverInfo.put("mail.smtp.debug", "true"); serverInfo.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory"); serverInfo.put("mail.smtp.socketFactory.fallback", "false"); serverInfo.put("mail.smtp.ssl.protocols", "TLSv1.2");//메일 서버와 SSL 통신을 하기 위한 설정 //네이버 사용자 인증 정보 auth = new Authenticator() { @Override protected PasswordAuthentication getPasswordAuthentication() { return new PasswordAuthentication("네이버 아이디","비밀번호"); //네이버 계정 정보 } }; }//end of instance block //이메일 전송 메소드 public void sendEmail(EmailDTO mailInfo) throws AddressException, MessagingException { //이메일 서버와의 세션 생성 Session session = Session.getInstance(serverInfo,auth); session.setDebug(true); //메일로 보낼 메세지 만들기 MimeMessage msg = new MimeMessage(session); msg.setFrom(new InternetAddress(mailInfo.getFrom()));//보내는 사람의 메일 주소 msg.addRecipient(Message.RecipientType.TO, new InternetAddress(mailInfo.getTo())); //받는 사람의 메일주소 msg.setSubject(mailInfo.getSubject());//제목 msg.setContent(mailInfo.getContent(),mailInfo.getFormat());//내용과 내용의 형식(MIME 타입) //메일보내기 Transport.send(msg); } }

mailForm.html(폼양식)

<!DOCTYPE html> <html> <meta charset="utf-8">   <table width="580" border="0" bordercolor="#DB0003" cellspacing="12" cstyle= "table-layout" align="center" style="margin:auto;"> <tbody> <tr> <td align="left" valign="top"> 제목

 내용:

   __CONTENT__

  </td> </tr> </tbody> </table>   </html>

sendEmailProcess.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%> <%@ page import="human.web.util.Conversion" %> <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> <jsp:useBean id="dto" class="human.web.dto.EmailDTO" /> <jsp:setProperty name="dto" property="*" /> <c:choose> <c:when test='${dto.format eq "text"}'> <jsp:setProperty name="dto" property="format" value="text/plain;charset=UTF-8" /> </c:when> <c:otherwise> <c:set var= "htmlContent" value="${Conversion.contentToHTML(pageContext.servletContext,dto.content)}" /> <jsp:setProperty name="dto" property="content" value="${htmlContent}" /> <jsp:setProperty name="dto" property="format"

value="text/html;charset=UTF-8" /> </c:otherwise> </c:choose> <jsp:useBean id="emailManager" class="human.web.common.EmailManager" /> <c:catch var="exception"> ${emailManager.sendEmail(dto)}
 </c:catch> <c:choose> <c:when test="${empty exception}"> <h3>이메일 전송 성공</h3> </c:when> <c:otherwise> <h3>이메일 전송 실패</h3> </c:otherwise> </c:choose>

sendEmail.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%> <!DOCTYPE html> <html> <head> <meta charset="UTF-8"> <title>SMTP 메일보내기</title> </head> <body> <h2>메일쓰기</h2> <form action="sendEmailProcess.jsp" method="post"> <table> <tr> <td>보내는 사람 : <input type="text" name="from" value=""></td> </tr> <tr> <td>받는 사람 : <input type="text" name="to" value=""></td> </tr> <tr> <td>제목 : <input type="text" name="subject" value=""></td> </tr> <tr> <td> 형식 : <input type="radio" name="format" value="text" checked>Text <input type="radio" name="format" value="html" >HTML </td> </tr> <tr> <td><textarea name="content" cols="60" rows="10"></textarea></td> </tr> <tr> <td><input type="submit" value="전송하기"></td> </tr> </table> </form> </body> </html>

75일차-정규표현식

java

2024/04/16 16:51 http://blog.naver.com/cy_109/223417432959

<< 정규표현식(regular expression) >>

- 일정한 패턴을 가진 문자열을 표현하기 위한 형식 언어
- 고유한 자바스크립트가 아니고 Perl에서 도입하여 사용
- (형식) /(시작기호)패턴/(끝기호)플래그

- 패턴에 사용하는 주요 기호/메타문자들
. : 임의의 문자 1개(문자,숫자,특수문자,공백문자 상관없음)
{m,n} : 바로 앞 패턴을 최소 m번, 최대 n번 가지는 문자열
{n} : 바로 앞 패턴을 n번 가지는 문자열
{n,} : 바로 앞 패턴을 최소 n번 이상 가지는 문자열
+ : 바로 앞 패턴을 최소 1번 이상 가지는 문자열
? : 바로 앞 패턴을 최대 1번 가지는 문자열
* : 바로 앞 패턴을 최소 0번 이상 가지는 문자열, {0,}와 같음
| : or의 의미
[] : 대괄호 내의 문자는 or로 동작함
[] 내의 -(하이픈) : 범위 지정
[] 내에서 ^ : not의 의미
[] 밖에서 ^ : 문자열의 시작 의미
$: 문자열의 끝 의미
() : 그룹화, 검색하는 패턴을 그룹으로 나눔, 여러 개의 패턴에 대해 검색할 수 있게 함
< 괄호 내에서 문자열이 조건에 만족하는지 체크할 수 있는 조건식 >
- 조건식은 여러 개를 나열할 수 있고 그런 경우 모든 조건에 만족해야 true 반환
(?=.*[조건]) : 조건에 해당하는 값이 대상 문자열에 있으면 인덱스 0을 반환
인덱스값을 반환해야 해당 정규표현식에 맞는 것으로 검증함
< 간략화한 표현 >
₩d : 숫자, [0-9]와 같음
₩D : 숫자가 아닌 문자
₩w : 알파벳, 숫자, 언더스코어(_)

₩W : 알파벳, 숫자, 언더스코어가 아닌 문자

₩s : 공백문자(space, tab, new line)

₩S : 공백문자가 아닌 문자

- 주요 플래그(패턴을 검색하는 방법 결정)

1) i(ignore case) : 대소문자를 구분하지 않음

2) g(global) : 문자열 전체 검색

3) m(multi-line) : 여러 줄 검색

- 정규표현식에서 제공하는 test() 메소드

(형식) 정규표현식.test(대상 문자열)

: 대상 문자열이 정규표현식에 맞는지 체크해서 맞으면 true, 그렇지 않으면 false 반환

- [주의사항] 정규표현식은 따옴표를 사용하지 않음

예문

<script> const str1 = "AaaBBbbbAa1*"; const regexp1 = /A{2,3}/i;//A나 a가 2-3개 이어져 있는 문자열 console.log("str1 검증결과: "+regexp1.test(str1)); const str2 = "AaaBBBbbbAa1*"; const regexp2 = /B{2}/;//B가 2개 이어져 있는 문자열 console.log("str2 검증결과: "+regexp2.test(str2)); const str3 = "AaaBBB홍bbbAa1*"; const regexp3 = /[가-힣]/;//한글이 있는 문자열 console.log("str3 검증결과: "+regexp3.test(str3)); const str4 = "~!@#$%^&*()_+=1"; const regexp4 = /[A-Za-z0-9가-힣]/;//영어대소문자, 숫자, 한글이 있는 문자열 console.log("str4 검증결과: "+regexp4.test(str4)); const str5 = "AaaBBBbbbAa1*"; const regexp5 = /c+/i;//대소문자를 구분하지 않고 c가 1개 이상 있는 문자열 console.log("str5 검증결과: "+regexp5.test(str5)); const str6 = "12345a"; const regexp6 = /[^0-9]/;//숫자가 아닌 문자가 있는 문자열 console.log("str6 검증결과: "+regexp6.test(str6)); const str7 = "https://"; const regexp7 = /^https/;//https로 시작하는 문자열 console.log("str7 검증결과: "+regexp7.test(str7)); const str8 = "www.naver.com"; const regexp8 = /com$/;//com으로 끝나는 문자열 console.log("str8 검증결과: "+regexp8.test(str8)); const str9 = "www.naver.com1"; const regexp9 = /(?=.*[0-9])/;//앞에 어떤 문자가 몇개가 오든 숫자가 있는 문자열이면 조건에 만족 console.log("str9 검증결과: "+regexp9.test(str9)); const str10 = "https://www.naver.com"; const regexp10 = /(^https)([A-Za-z0-9./:]+)(com$)/; console.log("str10 검증결과: "+regexp10.test(str10)); //숫자 1개 이상을 포함하고 8글자 이상 10글자 이하인 문자열에 대한 정규표현식 //특수문자:~!@#$%&()|= const regexp11 = /^(?=.*[0-9])[0-9A-Za-z~!@#$%&()|=]{8,10}$/; const str11 = "https5com"; console.log("str11 검증결과: "+regexp11.test(str11)); //비밀번호는 8글자 이상 16글자 이하로 영문자, 숫자, 특수문자를 1개 이상 포함해야 합니다. //위 조건을 체크할 수 있는 정규표현식을 작성하시오.(특수문자:~!@#$%^()+|=) const regexp12 = /^(?=.*[A-Za-z])(?=.*[0-9])(?=.*[~!@#$%^()+|=])[0-9A-Za-z~!@#$%^()+|=]{8,16}$/; const str12 = "https5!com"; console.log("str12 검증결과: "+regexp12.test(str12)); //전화번호는 010-숫자4자리-숫자4자리로 입력해야 합니다.

//위 조건을 체크할 수 있는 정규표현식을 작성하시오. const regexp13 = /^010-₩d{4}-₩d{4}$/; const str13 = "010-1111-1111"; console.log("str13 검증결과: "+regexp13.test(str13)); </script>

74일차

코딩

2024/04/15 16:42 http://blog.naver.com/cy_109/223416231997

자격증 시험 내일 10시에 원서접수

3. 2024년 정기 기사 제2회 필기시험 일정

구 분		기사·서비스·산업기사		
		지역 구분	원서접수 시작	원서접수 종료
		종목 구분	시험일	
원서접수		수도권 외 지방	4.16(화) 10:00	4.19(금) 18:00
		수도권(서울, 인천, 경기)	4.16(화) 14:00	
시험일		2시간이하 종목	5.17(금), 5.19(일)~5.22(수) [5일간]	
		2시간초과 종목	5.9(목)~5.10(금), 5.12(일)~5.16(목) 5.23(목)~5.28(화) [13일간]	
공통 사항	사전입력서비스	4.12(금) 14:00 ~ 4.15(월) 23:59		
	장소변경서비스	4.29(월) 10:00 ~ 4.30(화) 23:59		
	빈자리 접수	5.3(금) 10:00 ~ 5.4(토) 18:00		
	환불기간	100%	4.16(화) 10:00 ~ 4.19(금) 23:59	
		50%	4.20(토) 00:00 ~ 5.4(토) 23:59	
	응시자격 기준일	5.28(화)		
	합격(예정)자 발표	6.5(수) 09:00		
	응시자격서류 제출	5.9(목) ~ 6.17(월) * 온라인서류제출 : 원서접수시작일 ~ 6.14(금)까지		

※ 계좌 환불의 경우 은행별 전산 점검 등으로 23:20이후 환불 처리 불가

---

package human.java_ex.io_ex; import java.io.BufferedReader; import java.io.BufferedWriter; import java.io.FileReader; import java.io.FileWriter; import java.io.IOException; import java.io.InputStream; import java.io.OutputStream; import java.io.PrintWriter; public class IOEx01 { /* JAVA IO(입출력) -java.io 패키지 - 입력과 출력의 기준: 자바 프로그램 - 입력과 출력의 데이터 흐름: 스트림(단방향) - 가장 기본이 되는 입출력 스트림: InputStream, OutputStream - 추상클래스 - 자바 프로그램에서 입출력을 하려면 입출력 객체가 있어야 함 InputStream <= System.in OutputStream <= System.out - 입력객체의 주요 메소드: read(), read(byte[] buffer) - 출력객체의 주요 메소드: write(), write(byte[] buffer) - 입출력 단위: 1byte(바이트 단위), 2byte(문자 단위) - 입출력에 단위에 따른 입출력 객체 바이트 단위: InputStream, OutputStream 문자단위: Reader, Writer - 추상클래스 (객체를 얻

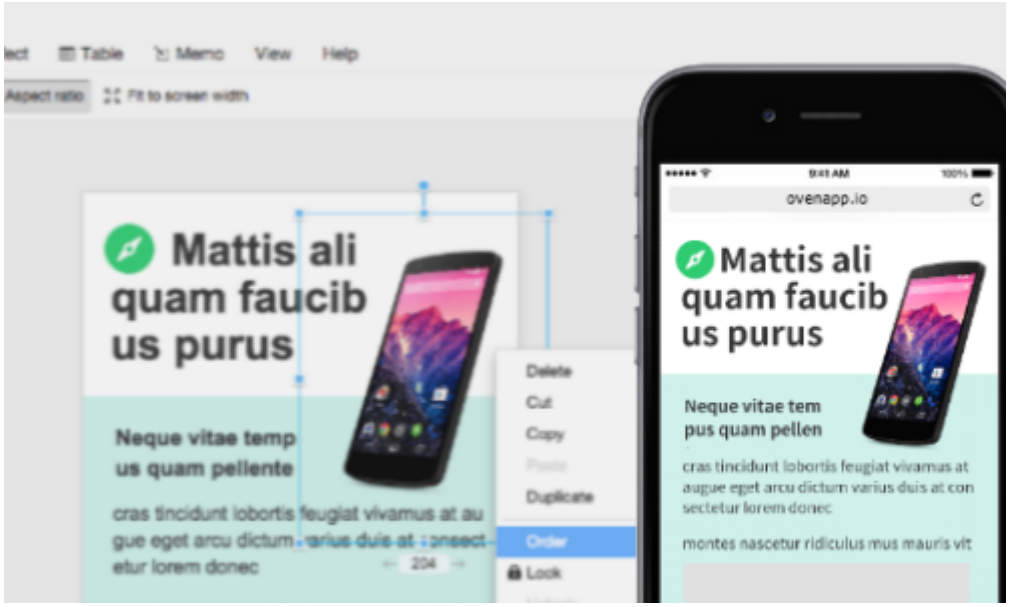
는 방법) InputStreamReader, OutputStreamWriter - 파일: File 클래스 - 파일객체: 디렉토리, 파일 - 시스템에 디렉토리와 파일 생성 방법 1. 파일 객체 생성: new File(디렉토리명 또는 파일명) 2. 기능을 수행하는 메소드 실행: mkdir(), mkdirs(), createNewFile() - 파일에 입출력을 할 수 있도록 지원하는 클래스: 일반 클래스 바이트 단위:FileInputStream, FileOutputStream 문자 단위: FileReader, FileWriter - 보조 스트림(기능 추가): 버퍼기능, 데이터타입, 객체, 편의기능 버퍼기능:Buffered- : BufferedInput/OutputStream, BufferedReader/Writer 데이터타입:Data- : DataInput/OutputStream 객체:Object- : ObjextInput/OutputStream 편의기능:Scanner, PrintWriter - 기본 입출력객체를 생성자의 매개변수로 해서 객체 생성 1. 기본 입출력 객체 생성 2. 보조 스트림 객체 생성 */ public static void main(String[] args) throws IOException, ClassNotFoundException { //가장 기본적인 입출력 객체: InputStream, OutputStream InputStream is = System.in; //입력매체: 키보드 OutputStream os = System.out; // 출력매체: 모니터 //바이트 단위 입출력하기 System.out.println("데이터 입력:"); byte b = (byte)is.read(); System.out.println("데이터 출력:"); os.write(b); os.flush();//출력객체에 저장된 데이터를 강제 출력시킴 //byte 배열을 이용한 데이터 입출력하기: 메소드 내부적으로 반복문 처리가 되어있음 byte[] buffer = new byte[1024]; System.out.println("데이터 입력:"); is.read(buffer); System.out.println("데이터 출력:"); os.write(buffer);//byte 배열에 저장된 모든 데이터 출력 os.write(buffer, 10, 20); 배열의 지정된 인덱스(10)부터 지정된 개수(20)만큼 출력 //문자단위 기본 입출력 객체: Reader, Writer Reader reader = new InputStreamReader(is); Writer writer = new OutputStreamWriter(os); // 보조 스트림 추가: 버퍼와 출력 편의성 추가 BufferedReader br = new BufferedReader(reader); BufferedWriter bw = new BufferedWriter(writer); PrintWriter pw = new PrintWriter(bw, true); //true: autoFlush 기능 실행 System.out.println("데이터 입력:"); String inputData = br.readLine(); //입력 데이터를 줄단위로 읽어서 문자열로 반환 System.out.println("데이터 출력:"); pw.println(inputData);// print()메소드 사용, 버퍼가 차지 않아도 자동으로 flush 기능 수행 //파일에 출력된 데이터 타입 그대로 입력값으로 가져오기: DataInput/OutputStream //프로젝트 폴더/test/test.dat 파일에 데이터를 출력하고 읽어오기 //test 디렉토리 생성하기 File dir = new File ("test"); dir.mkdir(); //test디렉토리 안에 test.dat파일을 생성하고 기본 데이터 타입으로 테이터 출력하기 //파일생성하기 File file = new File("test/test.dat"); file.createNewFile(); //파일 출력 객체는 해당 파일 없는 경우 파일울 생성해줌(디렉토리는 생성해주지 않음) FileOutputStream fos = new FileOutputStream("test/test.dat"); DataOutputStream dos = new DataOutputStream(fos); dos.writeInt(100); dos.writeDouble(100.25); dos.writeChar('A'); dos.writeUTF("홍길동"); System.out.println("기본타입 데이터 출력 완료"); FileInputStream fis = new FileInputStream("test/test.dat"); DataInputStream dis = new DataInputStream(fis); System.out.println("기본타입 데이터 입력 결과:"); System.out.println(dis.readInt()); System.out.println(dis.readDouble()); System.out.println(dis.readChar()); System.out.println(dis.readUTF()); //출력된 내용을 기본 타입 그대로 읽어올 때 출력된 순서대로 입력받아야 함 //객체 입출력하기: objectInput/OutputStream //프로젝트 폴더 test/test2.ser 파일에 Student객체를 출력하고 읽어오기 //Student클래스: name,year,handPhone을 필드로 가지고 매개변수 생성자를 //정의해서 객체 생성 시 필드가 초기화될 수 있도록 정의함 FileOutputStream fos = new FileOutputStream("test/test2.ser"); ObjectOutputStream oos = new ObjectOutputStream(fos); Student student = new Student("홍길동", 3, "010-1111-1111"); oos.writeObject(student); System.out.println("객체 출력 완료"); FileInputStream fis = new FileInputStream("test/test2.ser"); ObjectInputStream ois = new ObjectInputStream(fis); Student student2 =

(Student)ois.readObject(); System.out.println("입력객체 필드 확인:"); System.out.println("이름:"+student2.getName()); System.out.println("학년:"+student2.getYear()); System.out.println("번호:"+student2.getHandphone()); //File 클래스 //test/aaa/test3.txt 파일 생성하기 File dir2 = new File("test/aaa"); dir2.mkdir(); //디렉토리 생성하기 File file2 = new File("test/aaa/test3.txt"); file2.createNewFile(); //파일 생성하기 System.out.println("디렉토리와 파일 생성 완료"); //디렉토리 내에 있는 디렉토리와 파일 출력하기 //test 디렉토리 안에 있는 내용을 가져와서 디렉토리와 파일로 각각 출력하기 File dir3 = new File("test"); File[] files = dir3.listFiles();// 디렉토리 안에 있는 내용을 파일객체 배열로 반환
for(File file : files) { if(file.isDirectory()){//파일 객체가 디렉토리인지 확인하여 boolean 값 반환 System.out.println("디렉토리:"+file); }else { System.out.println("파일:"+file+"파일크기:"+file.length()+"byte"); } } System.out.println("출력 완료"); //파일에 데이터 출력하기: 파일 복사 - 텍스트 파일: FileReader/FileWriter //test/aaa/test3.txt 파일에 Student.java파일 출력하기 //src/main/java/human/java_ex/io_ex/Student.java FileReader fr = new FileReader("src/main/java/human/java_ex/io_ex/Student.java"); BufferedReader br = new BufferedReader(fr); FileWriter fw = new FileWriter("test/aaa/test3.txt", true);//true: 출력내용을 기존의 내용 뒤에 붙임 BufferedWriter bw = new BufferedWriter(fw); PrintWriter pw = new PrintWriter(bw,true); //true: autoFlush 기능 실행 String data = null; while((data=br.readLine()) != null) { pw.println(data); } System.out.println("파일 입출력 완료"); } }

개인 프로젝트 --쇼핑몰

스토리보드 만들기

Oven (ovenapp.io)



OvenApp.io

Oven(오븐)은 HTML5 기반의 무료 웹/앱 프로토타이핑 툴입니다. (카카오 제공)

ovenapp.io

화면설계를 위한 응용 시스템

카카오 오븐

73일차

java

2024/04/12 17:23 http://blog.naver.com/cy_109/223413616541

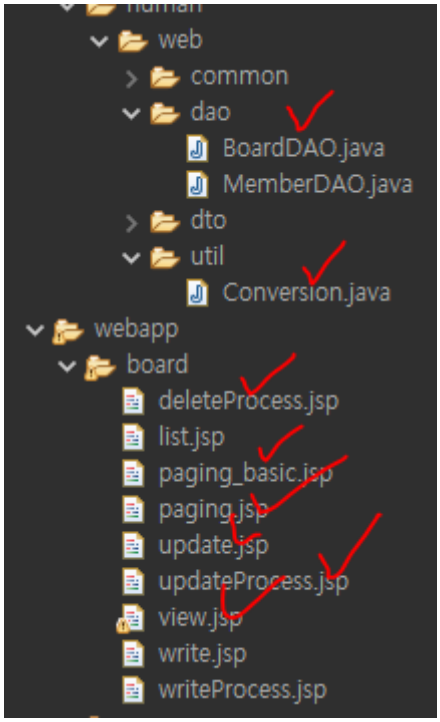
페이징 기능 만들기

글목록

총게시물: 60 제목 검색

번호	제목	작성자	조회수	등록일	첨부
1	테스트	채2	0	2024-04-12	
2	테스트	채2	0	2024-04-12	
3	테스트	채2	0	2024-04-12	
4	테스트	채2	0	2024-04-12	
5	테스트	채2	0	2024-04-12	
6	테스트	채2	0	2024-04-12	
7	테스트	채2	0	2024-04-12	
8	테스트	채2	0	2024-04-12	
9	테스트	채2	0	2024-04-12	
10	테스트	채2	0	2024-04-12	

1 2 3 4 5 다음페이지> >>



dao

//검색조건을 포함해서 모든 게시글 조회하기 public ArrayList<BoardDTO> getBoardList(SearchDTO dto){ ArrayList<BoardDTO> boardList = new ArrayList<>(); //제네릭을 이용해서 컬렉션 객체 생성 시 참조변수의 제네릭타입이 생성자의 제네릭타입과 같은 경우 //생성자의 제넥릭타입을 생략할 수 있음 try { if(dto.getSearchWord() != null) {//검색어로 검색한 경우 //검색영역을 체크하는 구문 String searchField = null; switch(dto.getSearchField()) { case "title": searchField = "title";break; case "content": searchField = "content";break; case "writer": searchField = "writer"; } String sql = "select * from tb_board " + " where board_status = 1 " + " and "+searchField+" like '%'||?||'%' " + " order by b_idx desc"; pstmt = conn.prepareStatement(sql); pstmt.setString(1, dto.getSearchWord());//검색어 세팅 }else {//검색어로 검색하지 않은 경우 String sql = "select * from tb_board " + " where board_status = 1 " + " order by b_idx desc"; pstmt = conn.prepareStatement(sql); } rs = pstmt.executeQuery();//조회된 결과를 ResultSet객체에 담음 while(rs.next()) {//rs.next():ResultSet객체의 BOF에서부터 시작해서 //저장된 데이터를 하나씩 확인해서 있으면 true 반환 BoardDTO bDto = new BoardDTO(); bDto.setB_idx(rs.getInt("b_idx")); bDto.setM_idx(rs.getInt("m_idx")); bDto.setWriter(rs.getString("writer")); bDto.setTitle(rs.getString("title")); bDto.setContent(rs.getString("content")); bDto.setPost_date(rs.getDate("post_date")); bDto.setRead_cnt(rs.getInt("read_cnt")); bDto.setOrigin_filename(rs.getString("origin_filename")); bDto.setSave_filename(rs.getString("save_filename")); boardList.add(bDto); } } catch (Exception e) { } return boardList; } //검색조건을 포함해서 총 게시물 수 조회하기 public int getTotalRows(SearchDTO dto){ int totalRows = 0;//총 게시물 수 조회 실패 시 결과값 try { if(dto.getSearchWord() != null) {//검색어로 검색한 경우 //검색영역을 체크하는 구문 String searchField = null; switch(dto.getSearchField()) { case "title": searchField = "title";break; case "content": searchField = "content";break; case "writer": searchField = "writer"; } String sql = "select count(*) from tb_board " + " where board_status = 1 " + " and "+searchField+" like '%'||?||'%' " + " order by b_idx

desc"; pstmt = conn.prepareStatement(sql); pstmt.setString(1, dto.getSearchWord());//검색어 세팅 }else {//검색어로 검색하지 않은 경우 String sql = "select count(*) from tb_board " + " where board_status = 1 " + " order by b_idx desc"; pstmt = conn.prepareStatement(sql); } rs = pstmt.executeQuery();//조회된 결과를 ResultSet객체에 담음 if(rs.next()) { totalRows = rs.getInt(1); } } catch (Exception e) { System.out.println("총 게시물 수 조회 중 예외 발생"); } return totalRows; } //조회수 증가시키기 public void updateRead_cnt(int b_idx) { try { String sql = "update tb_board set read_cnt = (read_cnt + 1) " + " where b_idx = ?"; pstmt = conn.prepareStatement(sql); pstmt.setInt(1, b_idx); pstmt.executeUpdate(); } catch (Exception e) { System.out.println("조회수 업데이트 중 예외 발생"); } } //게시글 가져오기 public BoardDTO getBoard(int b_idx) { BoardDTO dto = null; try { String sql = "select * from tb_board where b_idx = ?"; pstmt = conn.prepareStatement(sql); pstmt.setInt(1, b_idx); rs = pstmt.executeQuery(); if(rs.next()) {//조회된 결과값이 있는 경우 dto = new BoardDTO(); dto.setB_idx(rs.getInt("b_idx")); dto.setM_idx(rs.getInt("m_idx")); dto.setWriter(rs.getString("writer")); dto.setTitle(rs.getString("title")); dto.setContent(rs.getString("content")); dto.setOrigin_filename(rs.getString("origin_filename")); dto.setSave_filename(rs.getString("save_filename")); dto.setRead_cnt(rs.getInt("read_cnt")); dto.setPost_date(rs.getDate("post_date")); dto.setUpdate_date(rs.getDate("update_date")); } } catch (Exception e) { System.out.println("게시글 조회 중 예외 발생"); } return dto; } //게시글 수정하기 public int updateBoard(BoardDTO dto) { int result = 0; //글수정 실패 시 결과값 try { if(dto.getOrigin_filename() != null) {//첨부파일이 업로드된 경우 String sql="update tb_board set title=?,content=?,origin_filename=?, " + " save_file_name=?, update_date=sysdate " + " where b_idx=?"; pstmt = conn.prepareStatement(sql); pstmt.setString(1, dto.getTitle()); pstmt.setString(2, dto.getContent()); pstmt.setString(3, dto.getOrigin_filename()); pstmt.setString(4, dto.getSave_filename()); pstmt.setInt(5, dto.getB_idx()); }else {//첨부파일이 업로드되지 않은 경우 String sql="update tb_board set title=?,content=?, update_date=sysdate " + " where b_idx=?"; pstmt = conn.prepareStatement(sql); pstmt.setString(1, dto.getTitle()); pstmt.setString(2, dto.getContent()); pstmt.setInt(3, dto.getB_idx()); } result = pstmt.executeUpdate();//실행 시 적용된 행의 수 반환
} catch (Exception e) { System.out.println("게시글 수정 중 예외 발생"); } return result; } //게시글 삭제 처리하기:update public int deletePost(int b_idx) { int result = 0; //삭제 처리 실패 시 결과값 try { String sql = "update tb_board set board_status = -1 " + " where b_idx = ?"; pstmt = conn.prepareStatement(sql); pstmt.setInt(1, b_idx); result = pstmt.executeUpdate();//실행 시 적용된 행의 수 반환 } catch (Exception e) { System.out.println("게시글 삭제 중 예외 발생"); } return result; }

conversion.java(형변환 시켜주는 메소드)

package human.web.util; public class Conversion { //double형의 값을 소수점 이하 자리를 없애고 int값으로 변환하는 메소드 public static int doubleToInt(double d) { return (int)d; } //입력되는 아이디에 "아이디:"문자열을 추가해서 반환하는 메소드 public static String addStr(String member_id) { return "아이디: "+member_id; } }

deleteProcess.jsp (게시글 삭제 기능)

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%> <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> <c:set var="b_idx" value="${param.b_idx}" /> <c:set var="m_idx" value="${param.m_idx}" /> <jsp:useBean id="dao" class="human.web.dao.BoardDAO"/> <!-- 회원의 게시글인지 다시 한번 체크 후 글삭제 진행 --> <c:if test="${member.m_idx eq m_idx}"> <c:set var="result" value="${dao.deletePost(b_idx)}" /> <c:choose> <c:when test="${result eq 1}"> <c:redirect url="/board/list.jsp" context="${pageContext.request.contextPath}" /> </c:when> <c:otherwise> <script> alert("글삭제에 실패했습니다"); history.back(); </script> </c:otherwise> </c:choose> </c:if>

paging.jsp(글목록 페이지 넘기는 기능)

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%> <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> <!-- 페이지 네비게이션에서 페이지 시작번호: startPageNum --> <c:set var="startPageNum" value="${(pageBlock-1)*pages_per_block+1}" /> <!-- 페이지 네비게이션에서 페이지 끝번호: endPageNum --> <c:set var="endPageNum" value="${pageBlock*pages_per_block}" /> <!-- 첫번째 페이지 표시와 이전페이지 출력하기 --> <c:if test="${pageNum > pages_per_block}"> <<   <이전페이지 </c:if> <c:forEach var="i" begin="${startPageNum}" end="${endPageNum}" > <!-- 페이지 번호가 전체 페이지번호보다 작거나 같은 경우에만 출력되도록 조건문으로 처리함 --> <c:if test="${i le totalPageNum}" > ${i} </c:if> </c:forEach> <!-- 다음페이지와 마지막 페이지 표시 출력하기 --> <c:if test="${((rows_per_page*pages_per_block) lt totalRows) and (pageBlock ne lastPageBlock)}"> 다음페이지>   >> </c:if>

update(게시글 수정)

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%> <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> <c:set var="b_idx" value="${param.b_idx}" /> <jsp:useBean id="dao" class="human.web.dao.BoardDAO" /> <!-- 게시글 가져오기 --> <c:set var="dto" value="${dao.getBoard(b_idx)}" /> <!DOCTYPE html> <html> <head> <meta charset="UTF-8"> <title>글수정</title> <link rel="stylesheet" href="${pageContext.request.contextPath}/resources/css/update.css"> </head> <body> <form name="frm_update" action="updateProcess.jsp" method="post" enctype="multipart/form-data"> <table> <caption>글수정</caption> <tr> <th>작성자(닉네임)</th> <td> ${dto.writer} <input type="hidden" name="b_idx" value="${dto.b_idx}"> </td> </tr> <tr> <th>제목</th> <td> <input type="text" name="title" value="${dto.title}"> </td> </tr> <tr> <th>내용</th> <td> <textarea rows="10" cols="30" name="content">${dto.content}</textarea> </td> </tr> <tr> <th>첨부파일</th> <td> <input type="file" name="uploadFile"> 새로운 파일을 선택하면 이전 파일이 교체됩니다 </td> </tr> <tr> <td colspan="2" id="td_btn"> <input type="submit" value="수정하기"> <input type="reset" value="다시입력">

<input type="button" value="목록보기" onclick="location.href='list.jsp'"> </td> </tr> </table> </form> </body> </html>

updateProcess.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%> <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> <jsp:useBean id="fileProcess" class="human.web.common.FileProcess" /> <c:set var="dto" value="${fileProcess.fileProcess(pageContext.servletContext, pageContext.request)}" /> <jsp:useBean id="dao" class="human.web.dao.BoardDAO"/> <c:set var="result" value="${dao.updateBoard(dto)}" /> <c:choose> <c:when test="${result eq 1}"> <c:redirect url="/board/view.jsp?b_idx=${dto.b_idx}" context="${pageContext.request.contextPath}" /> </c:when> <c:otherwise> <script> alert("글수정에 실패했습니다"); history.back(); </script> </c:otherwise> </c:choose>

view.jsp(글 내용 작성)

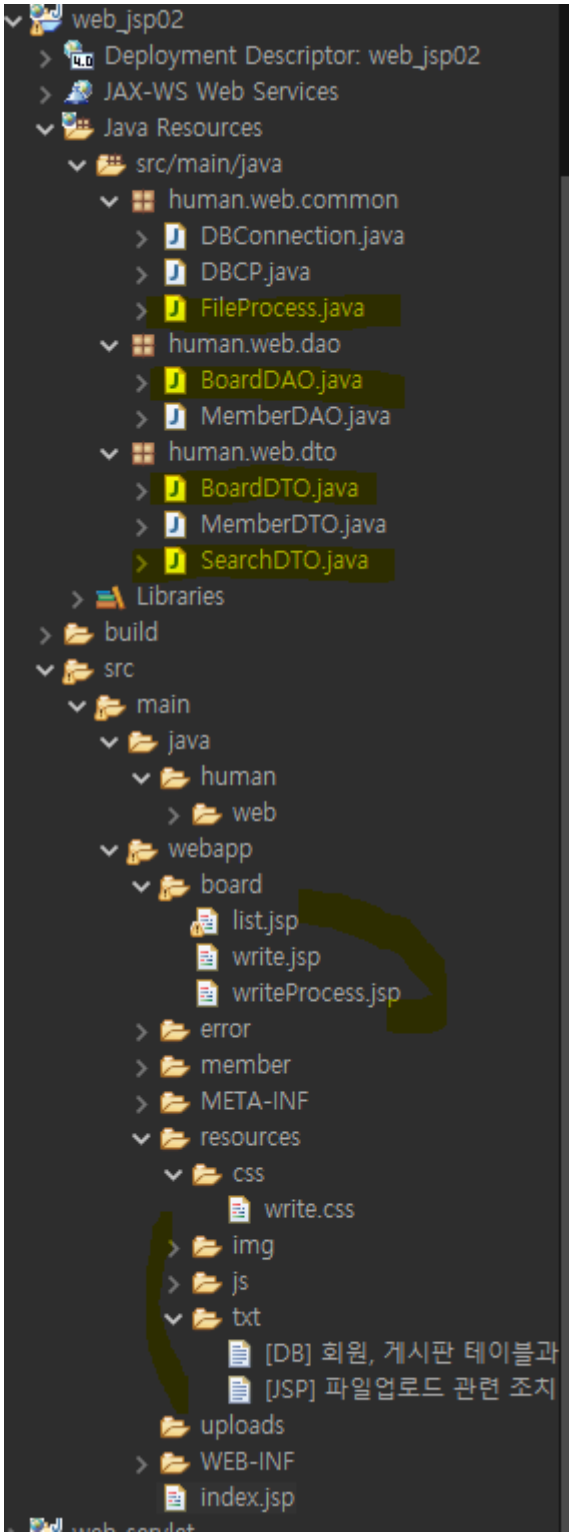
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%> <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> <c:set var="b_idx" value="${param.b_idx}" /> <jsp:useBean id="dao" class="human.web.dao.BoardDAO" /> <!-- 조회수 증가시키기 --> ${dao.updateRead_cnt(b_idx)}
 <!-- 게시글 가져오기 --> <c:set var="dto" value="${dao.getBoard(b_idx)}" /> <!DOCTYPE html> <html> <head> <meta charset="UTF-8"> <title>글내용</title> <link rel="stylesheet" href="${pageContext.request.contextPath}/resources/css/view.css"> <script> function deletePost(){ const ans = confirm("정말로 삭제하겠습니까?"); if(ans){ location.href="deleteProcess.jsp?b_idx=${dto.b_idx}&m_idx=${dto.m_idx}"; } } </script> </head> <body> <table> <caption>글내용</caption> <tr> <th>작성자(닉네임)</th> <td>${dto.writer}</td> </tr> <tr> <th>제목</th> <td>${dto.title}</td> </tr> <tr> <th>내용</th> <td><textarea cols="30" rows="5" disabled>${dto.content}</textarea></td> </tr> <tr> <th>첨부파일</th> <td> <img src="${pageContext.request.contextPath}/uploads/${dto.save_filename}" width="100px" height="70px"> ${dto.origin_filename} </td> </tr> <tr> <td colspan="2" id="td_btn"> <input type="button" value="수정하기" onclick="location.href='update.jsp?b_idx=${dto.b_idx}'"> <input type="button" value="삭제하기" onclick="deletePost();"> <input type="button" value="목록보기" onclick="location.href='list.jsp'"> </td> </tr> </table> </body> </html>

72일차

java

2024/04/11 15:42 http://blog.naver.com/cy_109/223412399461

회원가입 시스템에 게시판 기능 추가



파일 업로드 관련 조치

1. 파일 업로드를 지원하는 API 다운로드 및 추가하기
- https://www.servlets.com/cos/에서 cos-22.05.zip 다운로드 및 압축풀기
- WEB-INF/lib/cos.jar파일 추가

2. 프로젝트 폴더에 uploads 폴더 만들기
- 파일 업로드되는 폴더

3. <form>태그 속성 정의
- method="post"
- enctype="multipart/form-data"

4. <input type="file" name="uploadFile"> 정의

5. 사용자단에서 올린 게시물 정보는 모두 MultipartRequest객체에 저장됨

회원, 게시판 테이블과 시퀀스 코드
<< 회원 테이블 >> create table tb_member(m_idx number, --1.회원번호 member_id varchar2(30) unique not null, --2.아이디 member_pw varchar2(20) not null, --3.비밀번호 member_name varchar2(50) not null, --4.이름 nickname varchar2(30) not null, --5.닉네임 email varchar2(50) not null, --6.이메일 handphone varchar2(20) not null, --7.핸드폰번호 grade number(1) default 0, --8.등급(0:일반,1:관리자,2:슈퍼관리자) reg_date date default sysdate, --9.회원가입일 update_date date default sysdate, --10.회원정보 수정일 member_status number(1) default 1, --11.회원상태(1:유지,-1:탈퇴) constraint tb_member_pk primary key (m_idx)); create sequence member_seq minvalue 1 maxvalue 9999999 nocycle nocache noorder; << 게시판 테이블 >> create table tb_board(b_idx number, --1.게시글번호(기본키) m_idx number, --2.회원번호(참조키) writer varchar2(30) not null, --3.작성자 title varchar2(50) not null, --4.제목 content varchar2(1000) not null, --5.내용 orgin_filename varchar2(50), --6.원본파일명 save_filename varchar2(50), --7.저장파일명 read_cnt number(10) default 0, --8.조회수 post_date date default sysdate, --9.작성일 update_date date default sysdate, --10.수정일 board_status number(1) default 1, --11.게시물상태(1:유지, -1:삭제) constraint tb_board_pk primary key (b_idx), constraint tb_board_fk foreign key (m_idx) references tb_member (m_idx)); create sequence board_seq minvalue 1 maxvalue 9999999999 nocycle nocache noorder;

flieProcess.java

```
package human.web.common; import java.io.File; import java.text.SimpleDateFormat; import java.util.Date; import javax.servlet.ServletContext; import javax.servlet.http.HttpServletRequest; import com.oreilly.servlet.MultipartRequest; import human.web.dto.BoardDTO; public class FileProcess { // MultipartRequest객체 생성하기 private MultipartRequest getMultipartRequest( ServletContext application, HttpServletRequest request) { String uploadPath = application.getRealPath("/uploads");// 업로드된 파일을 저장할 디렉터리 int maxFileSize = 1024 * 1024 * 5;// 업로드 파일의 최대 크기(5 MB) String encoding = "UTF-8";// 업로드 파일의 인코딩 MultipartRequest mRequest = null; try { mRequest = new MultipartRequest(request, uploadPath, maxFileSize, encoding); // 정상적으로 MultipartRequest객체가 생성되면 //지정한 업로드 폴더로 지정한 인코딩으로 파일이 저장되고 // request객체가 받은 정보를 MultipartRequest객체가 저장하게 됨 } catch (Exception e) { System.out.println("MultipartRequest객체 생성 중 예외 발생"); } return mRequest; } // 파일 업로드와 업데이트에 대한 공통의 처리 메소드 public BoardDTO fileProcess(ServletContext application, HttpServletRequest request) { BoardDTO dto = new BoardDTO(); // 1. MultipartRequest객체 얻기 MultipartRequest mRequest = getMultipartRequest(application, request); // 2. 원본 파일명을 알아내고 날짜정보를 이용해서 uploads 폴더에 저장할 저장 파일명 만들기 String origin_filename = mRequest.getFilesystemName("uploadFile");// 원본파일명 String save_filename = null;// 저장파일명 if (origin_filename != null) {// 첨부파일을 업로드한 경우 // test.test.txt String ext = origin_filename.substring(origin_filename.lastIndexOf("."));// .확장자명 String now = new SimpleDateFormat("yyyyMMdd_HmsS").format(new Date()); save_filename = now + ext; // 3. 원본파일명을 저장파일명으로 바꾸기 String uploadPath = application.getRealPath("/uploads"); // 업로드된 파일을 저장할 디렉터리 File originFile = new File(uploadPath + File.separator + origin_filename); File saveFile = new File(uploadPath + File.separator + save_filename); originFile.renameTo(saveFile); // 4. BoardDTO의 필드에 세팅하기(원본파일명,저장파일명) dto.setOrigin_filename(origin_filename); dto.setSave_filename(save_filename); } // 4. BoardDTO의 필드에 세팅하기(회원번호,작성자,제목,내용) dto.setM_idx(Integer.parseInt(mRequest.getParameter("m_idx"))); dto.setWriter(mRequest.getParameter("writer")); dto.setTitle(mRequest.getParameter("title")); dto.setContent(mRequest.getParameter("content")); return dto; } }
```

BoardDAO.java

```
package human.web.dao; import java.util.ArrayList; import human.web.common.DBCP; import human.web.dto.BoardDTO; import human.web.dto.SearchDTO; public class BoardDAO extends DBCP { //게시글 입력하기 public int insertBoard(BoardDTO dto) { int result = 0;//게시글 입력 실패 시 결과값 try { if(dto.getOrigin_filename() != null) {//파일 업로드가 된 경우 String sql = "insert into tb_board " + " (b_idx,m_idx,writer,title,content,origin_filename,save_filename) " + " values(board_seq.nextval,?,?,?,?,?,?)"; pstmt = conn.prepareStatement(sql); pstmt.setInt(1, dto.getM_idx()); pstmt.setString(2, dto.getWriter()); pstmt.setString(3, dto.getTitle()); pstmt.setString(4, dto.getContent()); pstmt.setString(5,
```

dto.getOrigin_filename()); pstmt.setString(6, dto.getSave_filename()); }else {//파일 업로드가 되지 않은 경우 String sql = "insert into tb_board " + " (b_idx,m_idx,writer,title,content) " + " values(board_seq.nextval,?,?,?,?)"; pstmt = conn.prepareStatement(sql); pstmt.setInt(1, dto.getM_idx()); pstmt.setString(2, dto.getWriter()); pstmt.setString(3, dto.getTitle()); pstmt.setString(4, dto.getContent()); } result = pstmt.executeUpdate();//게시글 입력 처리 후 적용된 행의 수 반환 } catch (Exception e) { System.out.println("게시글 입력 중 예외 발생"); e.printStackTrace(); } return result; } //검색조건을 포함해서 모든 게시글 조회하기 public ArrayList<BoardDTO> getBoardList(SearchDTO dto){ ArrayList<BoardDTO> boardList = new ArrayList<>(); //제네릭을 이용해서 컬렉션 객체 생성 시 참조변수의 제네릭타입이 생성자의 제네릭 타입과 같은 경우 //생성자의 제넥릭 타입을 생략 할 수 있음 try { if(dto.getSearchWord() != null) { //검색어로 검색한 경우 //검색영역을 체크하는 구문 String searchField = null; switch(dto.getSearchField()) { case "title": searchField = "title"; break; case "content": searchField = "content";break; case "writer": searchField = "writer"; break; } String sql= "selecct * form tb_board" + "where board_status = 1" + "and"+searchField+"like '%'||?||'%'" + "order by b_idx desc"; pstmt = conn.prepareStatement(sql); pstmt.setString(1, dto.getSearchWord());//검색어 세팅 }else { //검색어로 검색하지 않은 경우 String sql= "selecct * form tb_board" + "where board_status = 1" + "order by b_idx desc"; pstmt = conn.prepareStatement(sql); } rs=pstmt.executeQuery();//조회된 결과를 resultSet 객체에 담음 while(rs.next()) {//rs.next():ResultSet객체의 BOF에서부터 시작해서 //저장된 데이터를 하나씩 확인해서 있으면 true 반환 BoardDTO bDto = new BoardDTO(); bDto.setB_idx(rs.getInt("b_idx")); bDto.setM_idx(rs.getInt("m_idx")); bDto.setWriter(rs.getString("writer")); bDto.setTitle(rs.getString("title")); bDto.setContent(rs.getString("content")); bDto.setPost_date(rs.getDate("post_date")); bDto.setRead_cnt(rs.getInt("read")); bDto.setOrigin_filename(rs.getString("origin_filename")); bDto.setSave_filename(rs.getString("save_filename")); boardList.add(bDto); } } catch (Exception e) { } return boardList; } }

BoardDTO.java

package human.web.dto; import java.util.Date; public class BoardDTO { private int b_idx;//게시글번호 private int m_idx;//회원번호 private String writer;//작성자 private String title;//제목 private String content;//내용 private String origin_filename;//원본파일명 private String save_filename;//저장파일명 private int read_cnt;//조회수 private Date post_date;//작성일 private Date update_date;//수정일 private int board_status;//게실물상태(1:유지,-1:삭제) public int getB_idx() { return b_idx; } public void setB_idx(int b_idx) { this.b_idx = b_idx; } public int getM_idx() { return m_idx; } public void setM_idx(int m_idx) { this.m_idx = m_idx; } public String getWriter() { return writer; } public void setWriter(String writer) { this.writer = writer; } public String getTitle() { return title; } public void setTitle(String title) { this.title = title; } public String getContent() { return content; } public void setContent(String content) { this.content = content; } public String getOrigin_filename() { return origin_filename; } public void setOrigin_filename(String origin_filename) { this.origin_filename = origin_filename; } public String getSave_filename() { return save_filename; } public void setSave_filename(String save_filename) { this.save_filename = save_filename; } public int getRead_cnt() {

return read_cnt; } public void setRead_cnt(int read_cnt) { this.read_cnt = read_cnt; } public Date getPost_date() { return post_date; } public void setPost_date(Date post_date) { this.post_date = post_date; } public Date getUpdate_date() { return update_date; } public void setUpdate_date(Date update_date) { this.update_date = update_date; } public int getBoard_status() { return board_status; } public void setBoard_status(int board_status) { this.board_status = board_status; } }

SearchDTO.java

package human.web.dto; public class SearchDTO { private String searchField; private String searchWord; public String getSearchField() { return searchField; } public void setSearchField(String searchField) { this.searchField = searchField; } public String getSearchWord() { return searchWord; } public void setSearchWord(String searchWord) { this.searchWord = searchWord; } }

list.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%> <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%> <!DOCTYPE html> <html> <head> <meta charset="UTF-8"> <title>글목록</title> </head> <body> <!-- 게시물 목록 가져오기: boardList --> <jsp:useBean id="dao" class="human.web.dao.BoardDAO" /> <!-- 사용자가 입력한 검색조건을 저장할 SearchDTO 생성 및 갑 세팅하기 --> <jsp:useBean id="searchDto" class="human.web.dto.SearchDTO" /> <jsp:setProperty name="searchDto" property="*" /> <!-- 검색조건을 포함해서 게시글 목록 가져오기 --> <c:set var="boardList" value="${dao.getBoardList(searchDto)}" /> <!--검색 폼--> <form> <table> <caption>글목록</caption> <tr> <td> 총게시물:</td> <td><select name="searchField"> <option value="title">제목</option> <option value="content">내용</option> <option value="writer">작성자</option> </select> <input type="text" name="searchWord"></td> </tr> </table> </form> <!-- 글목록 테이블 --> <table> <tr> <th>번호</th> <th>제목 </th> <th>작성자</th> <th>조회수</th> <th>등록일</th> <th>첨부</th> </tr> <!-- 글목록 내용 --> <c:choose> <c:when test="${empty boardList}"> <tr> <td colspan="6">등록한 게시물이 없습니다.</td> </tr> </c:when> <c:otherwise> <c:forEach var="board" items="${boardList}"> <tr> <td> <!-- 번호 --> ${status.count} </td> <td> <!-- 제목 --> ${board.title} </td> <td> <!-- 작성자 --> ${board.writer} </td> <td> <!-- 조회수 --> ${board.read_cnt} </td> <td> <!-- 등록일 --> ${board.post_date} </td> <td> <!-- 첨부 --> <!-- 첨부파일이 있을 경우 첨부파일 이미지를 출력되도록 함 --> <c:if test="${not empty board.origin_filename}"> <img src="${pageContext.request.contextPath }/resources/img/download.png" width="15px" height="17px"> </c:if> </td> </tr> </c:forEach> </c:otherwise> </c:choose> </table> </body> </html>

write.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%> <!DOCTYPE html> <html> <head> <meta charset="UTF-8"> <title>글등록</title> <link rel="stylesheet" href="${pageContext.request.contextPath}/resources/css/write.css"> </head> <body> <form name="frm_write" action="writeProcess.jsp" method="post" enctype="multipart/form-data"> <table> <caption>글등록</caption> <tr> <th>작성자(닉네임)</th> <td> <input type="text"

value="${member.nickname}" disabled> <input type="hidden" name="writer" value="${member.nickname}"> <input type="hidden" name="m_idx" value="${member.m_idx}"> </td> </tr> <tr> <th>제목</th> <td> <input type="text" name="title" > </td> </tr> <tr> <th>내용</th> <td> <textarea rows="10" cols="30" name="content"></textarea> </td> </tr> <tr> <th>첨부파일</th> <td> <input type="file" name="uploadFile"> </td> </tr> <tr> <td colspan="2" id="td_btn"> <input type="submit" value="작성완료"> <input type="reset" value="다시입력"> <input type="button" value="목록보기" onclick="location.href='list.jsp'"> </td> </tr> </table> </form> </body> </html>

writeProcess.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%> <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> <jsp:useBean id="fileProcess" class="human.web.common.FileProcess" /> <c:set var="dto" value="${fileProcess.fileProcess(pageContext.servletContext, pageContext.request)}" /> <jsp:useBean id="dao" class="human.web.dao.BoardDAO"/> <c:set var="result" value="${dao.insertBoard(dto)}" /> <c:choose> <c:when test="${result eq 1}"> <c:redirect url="/board/list.jsp" context="${pageContext.request.contextPath}" /> </c:when> <c:otherwise> <script> alert("글등록에 실패했습니다"); history.back(); </script> </c:otherwise> </c:choose>

71일차

java

2024/04/09 17:28 http://blog.naver.com/cy_109/223410613151

o <c:import>: <jsp:include>액션태그의 기능을 제공하는 태그(JSP외 다른 웹페이지도 호출 가능)

(예)<c:import url="productURL">
<c:param name="produxt" value="TV">
<c:param name="price" value="450000">
</c:import>

o <c:url>: <c:set>처럼 변수 선언에 사용되고 URL을 쉽게 다룰 수 있는 방법을 제공하는 태그
value 속성: 컨텍스트 경로를 기본적으로 포함하고 있으므로 컨텍스트 이하의 경로를 붙이면 됨
(예)
<c:url var="sumURL" value="add.jsp">
<c:param name="num1" value="100">
<c:param name="num2" value="200">
</c:url>
<c:redirect url="${sumURL}"/>

o <c:out>:
표현식과 EL과 동일한 역할
출력내용에 포함된 HTML태그를 해석해서 적용할 수 있는 escapeXml속성이 있음
escapeXml속성을 false로 지정하면 HTML 태그를 해석하여 마크업이 적용된 상태로 출력함
출력내용이 null일 경우 기본값을 지정해서 출력할 수 있는 default속성이 있음
빈 문자열은 값으로 해석해서 default 값이 출력되지 않음

(예)
<c:out value="JSTL의 태그 라이브러리 중 <core>라이브러리 내용입니다"/>
<c:out vlaue="${str}" default="NO Data"/>

예시 코드들

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%> <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> <!DOCTYPE html> <html> <head> <meta charset="UTF-8"> <title>JSTL: core 예제</title> </head> <body> <!-- core 태그 라이브러리 예제 목록 *** JSTL core 태그 라이브러리를 이용하기 위해서 taglib 지시자 추가 1. set태그, remove태그 2. 조건문 태그: if, choose-when-otherwise 3. 반복문 태그: forEach, forTokens 4. 페이지 이동 태그: import, redirect 5. url 태그 6. out 태그 7. catch 태그 --> <!-- 1. set태그, remove태그 --> <!-- 기본 타입의 변수 선언 및 초기화 --> <c:set var="num1" value="123" scope="page" /> <c:set var="num2" value="${ num1 mod 5 }" /> <c:set var="num3">500</c:set> 변수 num1: ${ num1 }
 변수 num2: ${ num2 }
 변수 num3: ${ num3 }
 <!-- set태그로 변수 선언 및 초기화시 scope속성을 지정하지 않으면 기본적으로 page영역에 저장됨 --> <!-- 참조타입의 변수 선언 및 초기화 --> <%@ page import="human.jsp_ex.util.*, java.util.*" %> <c:set var="now" value="${ Data.getDate() }" /> 현재 날짜와 시간: ${ now }
 <c:set var="hero" value='${ Data.getPerson("이순신", 30) }' /> <c:set var="hero2" value='${ Data.getPerson("강감찬", 50) }' /> <c:set var="hero3" value='${ Data.getPerson("김유신", 40) }' /> 영웅의 이름과 나이: ${ hero.name }, ${ hero.age }
 <!-- List 컬렉션 변수 선언 및 초기화 --> <c:set var="heroList" value='${ Data.getPersonList(hero, hero2, hero3) }' /> 영웅의 이름과 나이: ${ heroList[0].name }, ${ heroList[0].age }
 영웅의 이름과 나이: ${ heroList[1].name }, ${ heroList[1].age }
 영웅의 이름과 나이: ${ heroList[2].name }, ${ heroList[2].age }
 <!-- 선언된 변수 삭제하기: remove 태그 --> <c:remove var="hero" /> 영웅의 이름과 나이: ${ hero.name }, ${ hero.age }
 <hr> <!-- 2. 조건문 태그: if, choose-when-otherwise - var 속성에 test 속성의 결과값을 저장할 수 있음 - test 속성에 오는 값이 EL이 아닌 경우 false를 반환하고 대소문자 구분 없이 true인 경우 true 반환함 - test 속성에 오는 EL 양쪽에 공백이 있는 경우 false를 반환함 --> <!-- 홀수 짝수 판단하기 --> <c:if test="${ 123 mod 2 eq 0 }" var="result"> 123은 짝수이다
 </c:if> <c:if test="${ not result }"> 123은 홀수이다
 </c:if> <c:if test="홍길동" var="result2" /> <c:if test=" ${true} " var="result3" /> test 속성에 EL이 아닌 값이 오는 경우 결과값: ${result2}
 test 속성에 오는 EL 양쪽에 공백이 있는 경우 결과값: ${result3}
 <!-- 조건문: choose - when - otherwise --> <!-- 검색어가 있는 경우와 검색어가 없는 경우 --> <!-- 검색 폼 --> <div> <form><!-- action 속성이 없으면 현재 JSP페이지를 호출하여 입력값을 보냄 --> <select name="searchField"> <option value="title">제목</option> <option value="content">내용</option> <option value="writer">작성자 </option> </select> <input type="text" name="searchWord" id="searchWord"> <input type="submit" id="search_btn" value="검색"> </form> </div> <c:choose> <c:when test="${not empty param.searchWord}"> 검색영역: ${ param.searchField }, 검색어: ${param.searchWord}
 </c:when> <c:otherwise> 검색어가 없습니다
 </c:otherwise> </c:choose> <hr> <!-- 3. 반복문 태그: forEach, forTokens - forEach 태그: 일반 for문, forEach문 모두 지원 - forEach의 varStatus속성: 제어변수 var의 상태에 대한 값을 여러가지 속성(index, count, current, first, last)으로 반환함 - forTokens: for문 + StringTokenizer 클래스 기능 --> <!-- 일반 for문: forEach 태그 1에서 100까지 정수 중 홀수의 합 출력하기 --> <c:forEach var="i" begin="1" end="100" step="1"> <!--(mod 나머지 연산자) i랑 2를 뺀 나머지가 (eq 값이 같은지 비교)1 인지 확인--> <c:if test="${i mod 2 eq 1}"> <c:set var="sum" value="${sum+i}" /> </c:if> </c:forEach> 1부터 100까지 홀수의 합: ${sum}
 <!-- 2단부터 3단까지 구구단 출력

하기 : 이중 forEach문을 사용 - varStatus속성 이용 각 단의 첫번째 행을 출력하기 전에 몇 단인지 출력하도록 하고 마지막 행을 출력한 후에 구분선(---------)을 출력하도록 하기 --> <c:forEach var="dan" begin="2" end="3"> <c:forEach var="i" begin="1" end="9" varStatus="status"> <c:if test="${status.first}"><!-- 각 단의 첫번째 행을 출력하기 전 --> ${dan} 단 </c:if> ${dan} X ${i} = ${dan*i}
 <c:if test="${status.last}"><!-- 마지막 행을 출력한 후 --> ----------
 </c:if> </c:forEach>
 </c:forEach> <!-- 확장 for문: forEach 태그 배열에 저장된 이름들을 콤마(,)로 구분해서 출력하되 마지막 값에는 콤마를 붙이지 않고 출력하기 ** Data클래스에 문자열 입력값을 받아서 문자열 배열로 반환하는 getStringArray()메소드를 정의하고 이것을 EL에서 이용하여 items에 값으로 입력함 --> <c:set var="names" value='${Data.getStringArray("홍길동", "이순신", "강감찬")}' /> <c:forEach var="name" items="${names}" varStatus="status"> ${name}<c:if test="${not status.last}">, </c:if> </c:forEach> <!-- List객체에 저장된 Person 객체의 이름과 나이 출력하기 --> <c:forEach var="person" items="${heroList}" > 이름: ${person.name}, 나이: ${person.age}
 </c:forEach> <!-- forTokens 태그를 이용해서 문자열을 토큰으로 출력하기 각 문자열을 #을 이용해서 구분되어지도록 하고 마지막에는 #을 붙이지 않고 출력하기 --> <c:set var="subjects" value="Java, HTML, CSS, Javascript, SQL, Servlet, JSP" /> <c:forTokens var="subject" items="${subjects}" delims="," varStatus="status"> ${subject}<c:if test="${not status.last}"># </c:if> </c:forTokens>

<hr> <!-- 4. 페이지 이동 태그: import, redirect import 태그: url속성의 ex07_otherPage.jsp의 실행결과를 var속성의 contents 변수에 저장함 --> <c:set var="bookName" value="JSP프로래밍" scope="request"/> <c:import var="contents" url="ex07_otherPage.jsp"> <c:param name="category1" value="컴퓨터/인터넷" /> <c:param name="category2" value="웹프로그래밍" /> </c:import> contents 변수에 저장된 내용:
 ${contents} <p> <!-- redirect 태그: response 객체의 sendRedirect()메소드와 같은 기능을 하는 태그 context 속성에 컨텍스트 경로를 지정하면 url 속성에는 그 이하의 경로를 지정해야 함 --> <%-- <c:redirect url="ex07_otherPage.jsp" > <c:param name="category1" value="컴퓨터/인터넷" /> <c:param name="category2" value="웹프로그래밍" /> </c:redirect> --%> <%-- <c:redirect url="/jsp_ex/ex07_otherPage.jsp" context="${pageContext.request.contextPath}"> <c:param name="category1" value="컴퓨터/인터넷" /> <c:param name="category2" value="웹프로그래밍" /> </c:redirect> --%> <hr> <!-- 5. url 태그: url을 변수로 저장해서 사용할 수 있게 함 value 속성: 컨텍스트 경로를 기본적으로 포함하고 있으므로 컨텍스트 이하의 경로를 붙이면 됨 --> <c:url var="url" value="/jsp_ex/ex07_otherPage.jsp" /> ex07_otherPage.jsp 바로가기 <p> <c:url var="main" value="../index.jsp" /> 메인페이지 <hr> <!-- 6. out 태그 - 표현식과 EL과 동일한 역할 - 출력내용에 포함된 HTML태그를 해석해서 적용할 수 있는 escapeXml속성이 있음 escapeXml속성을 false로 지정하면 HTML 태그를 해석하여 마크업이 적용된 상태로 출력함 - 출력내용이 null일 경우 기본값을 지정해서 출력할 수 있는 default속성이 있음 빈 문자열은 값으로 해석해서 default 값이 출력되지 않음 --> <c:set var="iTag"> i 태그는 기울임을 표현하다. </c:set> 표현식이나 EL과 동일한 역할을 하는 출력: <c:out value="${iTag}" />
 출력내용에 포함된 HTML 태그를 해석해서 출력: <c:out value="${iTag}" escapeXml="false" />
 출력내용이 null인 경우 default 속성값 출력: <c:out value="${parm.test}" default="테스트 내용"/>
 출력내용이 빈 문자열인 경우의 출력: <c:out value=""

default="테스트 내용"/>
 <hr> <!-- 7. catch 태그: 예외 발생 구문에서 예외를 받아서 var 속성에 저장함 --> <c:catch var="e"> <% int result = 100/0; %> </c:catch> <!-- 예외처리 구문 정의 --> <c:if test="${not empty e}" > 예외 발생: ${e} </c:if> </body> </html>