8_23_2022__Web_Regular Expression

context path를 얻어내는 두가지 방법

- java영역 : HttpServletRequest 객체의 .getContextPath()메소드를 호출하면 conttext path 가 문자열로 리턴도니다.
- jsp-html 영역 : EL을 이용하면 클라이언트 웹브라우저에 출력할 수 있다. ${pageContext.request.contextPath}

파일을 업로드 한다고 생각해 보면 업로드 되는 파일의 정보는

1. 파일의 이름
2. 실제 파일 데이터(byte 알갱이)
3. 파일의 크기

파일 업로드 폼 작성법

1. method="post"
2. enctype="multipart/form-data"
3. <input type="file" />

   enctype="multipart/form-data" 가 설정된 폼을 전송하면
   폼전송된 내용을 추출할때 HttpServletRequest 객체로 추출을 할수 없다
   MultipartRequest 객체를 이용해서 추출해야 한다.

▼ 코드

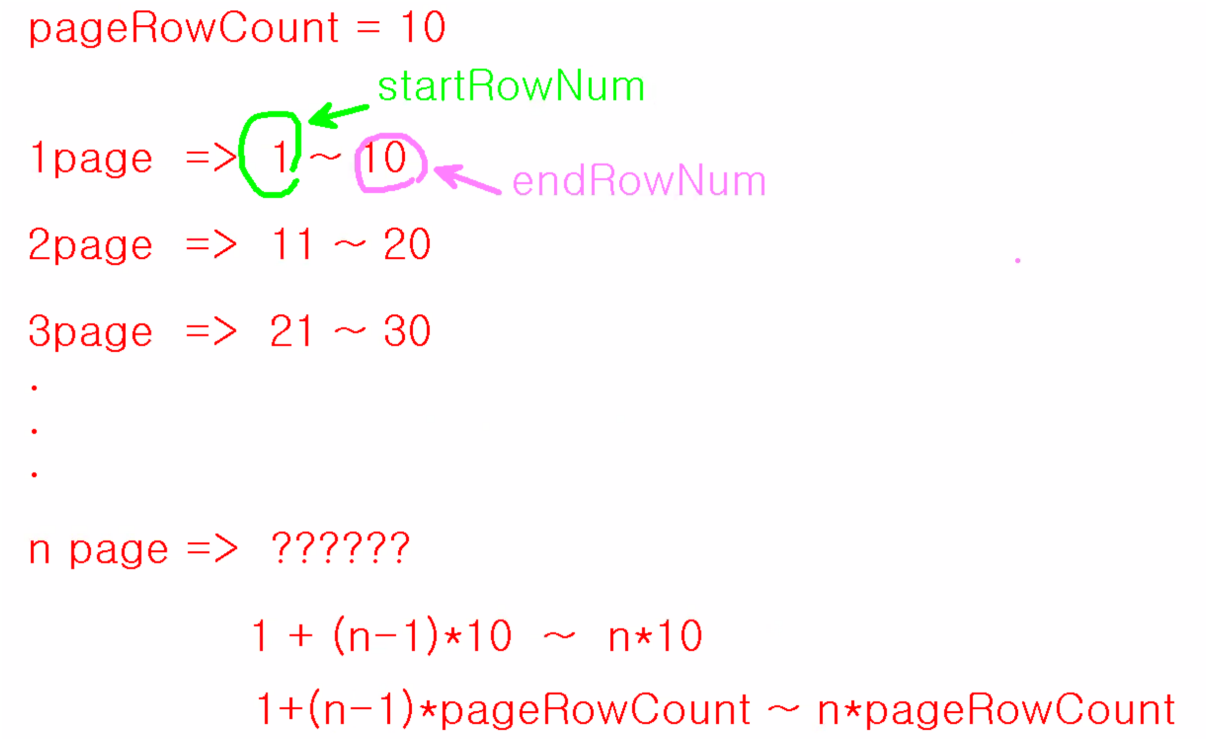
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>/file/private/upload_form.jsp</title>
</head>
<body>
```

```
   <div class="container">
     <h1>파일 업로드 폼 입니다.</h1>
     <!--
       파일 업로드 폼 작성법
       1. method="post"
       2. enctype="multipart/form-data"
       3. <input type="file" />
       - enctype="multipart/form-data" 가 설정된 폼을 전송하면
         폼전송된 내용을 추출할때 HttpServletRequest 객체로 추출을 할수 없다
        MultipartRequest 객체를 이용해서 추출해야 한다.
      -->
     <form action="upload.jsp" method="post" enctype="multipart/form-data">
       <div>
         <label for="title">제목</label>
         <input type="text" name="title" id="title"/>
       </div>
       <div>
         <label for="myFile">첨부파일</label>
         <input type="file" name="myFile" id="myFile"/>
       </div>
       <button type="submit">업로드</button>
     </form>
   </div>
 </body>
 </html>
```

페이징 처리 하는 방법

1. 정렬하기
2. 행번호 붙이기
3. 원하는 범위의 행 번호만 select 하기

```
SELECT *
FROM
   (SELECT result1.*, ROWNUM AS rnum
   FROM
      (SELECT num, writer, title, orgFileName
      FROM board_file
      ORDER BY num DESC) result1)
WHERE rnum BETWEEN 11 AND 20
```



정규 표현식 (Regular Expression)

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/12b157d4-deec-4c4f-9e6a-fa7fe2e9c6c6/javascript%EC%A0%95%EA%B7%9C%EC%8B%9D.pdf

특정한 규칙을 가진 문자열의 집합을 표현하는데 사용하는 형식 언어이다.

정규표현식을 이해하면 정규 표현식 객체를 활용해서 아래와 같은 작업을 할 수가 있다.

- 어떤 문자열에 특정 pattern의 문자열혹은 문자열이 있는지 여부를 알 수가 있다.
- 어떤 문자열에서 특정 pattern의 문자 혹은 문자열을 추출 할 수가 있다.
- 어떤 문자열이 특정 pattern과 정확히 일치하는지 여부를 알 수가 있다.
- 예)
  - 어떤 문자열에 특수문자가 포함되어 있는지 여부
  - 어떤 문자열이 이메일 형식과 일치하는지 여부
  - 어떤 문자열이 전화번호 형식과 일치하는지 여부

    - 어떤 문자열이 영어 대소문자와 숫자로만 이루어져 있는지 여부
    - 등 특정 패턴을 비교 확인할 수 있다.

- 빨간색은 검증대상 초록색은 정규 표현식
- 찾아졌다면 하이라이트 됨
- javascript에서 양쪽을 / 로 감싸서 만든다


```
let str="Hello, world!
let reg=/Hello/;
let reg2=/hello/;
```

- 정규표현식에 몇가지 기능이 생긴다.

```
reg.test(str)
//결과 true ->str에 Hello 가 들어있다
reg2.test(str)
//결과 false ->str에는 소문자로시작하는 hello가 있지 않다.
```

- 정규표현식을 만드는 다른 방법 (/를 사용하는게 쉽다)

```
let reg3= new regExp("Hello")
reg3.test(str)
```

- 어떤 문자는 특별한 의미를 가지고 있다
  ^는 매칭할 문자열의 시작을 의미하고 $는 매칭할 문자열의 끝을 의미한다.
- 특별한 의미를 가지고 있는 문자열의 literal 값이 필요하다면 역슬래쉬 \ 를 앞에 붙여줘야 한다.
- . 점은 모든 문자를 의미한다.(한 글자)
- . 점의 literal 값이 필요하다면 역시 역슬래시 \ 가 필요하다
- [ ] 대활호 안에는 매칭될 수 있는 문자의 목록을 넣는다.(한 글자) 순서는 중요하지 않다.
    - . 점은 아무 문자나 한글자를 선택하고 [ ]괄호는 후보군 목록 중 한글자를 선택한다.
- 문자의 범위는 [-] 문법으로 나타낼 수 있다 [ㄱ-ㅎ] ← ㄱ부터 ㅎ까지

- [ ] ⇒ 문자 클래스
  문자 클래스 안에 [^abc] 처럼 첫문자로 ^가 있다면 a,b,c각각은 매칭하지 않을 문자 목록이 된다.
- 문자열을 교차 매칭 시키려면 소괄호 안에 | 로 구분해서 문자열을 나열하면 된다.
  (문자열1|문자열2|문자열3) ← 문자열1 또는 문자열2 또는 문자열3
- 수량자(Quantifiers) : *, +, ?
  문자가 몇 번 올 수 있는지 정의한다.
  - * ← 0번이상(없어도 되고 여러개가 있어도 된다.)
    겨울에 눈이 많이오면 0친다 → 0번 이상
  - + ← 1번이상(반드시 있어야 하면 여러개가 있어도 된다.)
    양수가 연상된다 → 1번 이상
  - ? ← 0번 또는 1번(없어도 되고 있다면 오직 한개만 허용)
    이진수는 0과 1로 구성된다 →0111101 →숫자몇> → ?????모름
- { } 는 정확한 문자의 반복횟수를 정의한다
  - {m} 는 m번 반복
  - {m,n}는 최소 m번 최대 n번 반복
  - {m, }는 최소 m번 반복
  - * 는 {0, }와 같다.
  - +는 {1,}와 같다.
  - ? 는 {0,1}와 같다.

```
let reg1=/hello/;
//hello
let reg2=/^kim/;
//kim이 맨 앞에 있는
let reg3=/gura$/
//gura가 맨 뒤에 있는
let reg4=/\^\&/;
//^ &
let reg5=/.../;
//아무거나 세글자
let reg6=/\.\./;
//문자열 ..
let reg7=/[a-zA-Z0-9]/;
//소문자a부터z 대문자A부터Z 숫자0부터9 아무거나
let reg8=^[a-zA-Z0-9]&/;
```

```
//소문자로 시작하는거 부터 숫자로 끝나는거 아무거나
let reg9=
let result1=reg7.test("ABCabc@@@"); //true or false?
//flase
let result2=reg8.test("abc@@@"); //true or false?
//false
let result3=reg8.test("abc"); //true or false
//false
let result4=reg9.test("abc"); //true or false
//ture
```

- 동치 관계(대소문자는 반대 표현식)
  - [a-zA-Z0-9] == [\w]
  - [^a-zA-Z0-9] == [\W]
  - [0-9] == [\d]
  - [^0-9] == [\D]

/users/signup_form.jsp

```
<small class="form-text text-muted">영문자 소문자로 시작하고 5글자~10글자 이내로 입력하세요</small>

//영문자 소문자로 시작하고 5글자~10글자 이내
^[a-z],{4,9}$
```