

- 흐름

- [웹페이지 접근 요청] 사용자가 시작페이지(글 목록) URL로 접근 시도
  - a. List Controller(servlet) :: 사용자의 request를 받아 service 기능 구현한 뒤, 해당 request를 list.jsp에 전달 \*forward 방식(db에서 받아 request에 저장한 내용 그대로 전달)
  - b. list.jsp :: 글 목록 페이지
- [글 작성] 사용자가 글작성 버튼 클릭
  - a. FormController(servlet) :: 사용자의 request를 받아 service 기능 구현한 뒤, 해당 request를 form.jsp에게 전달. \*forward 방식?
  - b. form.jsp :: 글 작성 페이지. 작성완료후 write controller에게 request 전달
  - c. Write Controller(servlet) :: request 받아서 새 글을 db에 저장 \*sendredirect 방식(새로고침 때마다 해당 글 추가 방지)
  - d. List Controller(servlet) :: 새 db 받아서 request에 담아 list.jsp에 전달
  - e. list.jsp :: 글 목록 페이지(갱신된 모습)
- [글 수정] 사용자가 글 번호 클릭
  - a. EditFormController(servlet) :: 사용자의 request를 받아 service 구현한 뒤, request를 editForm.jsp에 전달
  - b. editForm.jsp :: 해당 글의 수정 페이지. 수정완료후 Edit controller 에게 request를 전달
  - c. EditController(servlet) :: request 받아서 수정된 글을 db에 저장 \*sendredirect방식(새로고침할 때마다 추가 변경 방지)
  - d. List Controller(servlet) :: 새 db 받아서 request에 담아 list.jsp에 전달
  - e. list.jsp :: 글 목록 페이지(갱신된 모습)
- [글 삭제] 사용자가 글 번호 클릭
  - a. EditFormController(servlet) :: 사용자의 request를 받아 service 구현한 뒤, request를 editForm.jsp에 전달
  - b. editForm.jsp :: 해당 글의 수정 페이지. 삭제완료후 Del controller 에게 request를 전달
  - c. DelController(servlet) :: request 받아서 db에 있는 데이터 삭제 \*sendredirect방식(새로고침할 때마다 추가 변경 방지)
  - d. List Controller(servlet) :: 새 db 받아서 request에 담아 list.jsp에 전달
  - e. list.jsp :: 글 목록 페이지(갱신된 모습)

## 1. 글 수정

- 수정기능 의사코드
  - 1) 작성자, 날짜는 fix  
내용은 띄워주되 수정 가능하게끔
  - 2) 수정버튼
  - 3) 수정 누르면 prompt로 글 비밀번호 입력창 띄워서 맞으면 수정완료, 아니면 취소
- 코드 흐름
  - 글 번호를 함께 보내기. get방식으로 뒤에 파라미터와 값 보내주기.  
상대편에서는 getparameter(파라미터)로 값 받기
  - EditFormController (servlet) 파일 생성  
\*scope 객체: 값을 담는다.  
담을 수 있는 종류 page, request\*, session\*, application

모두 `setAttribute()`, `getAttribute()`를 가진다.

page: 현재 페이지에서만 유지

request: 클라이언트의 요청을 객체로 만든 것.

해당 요청에 대한 응답을 보내기 전까지 유지

session: http 프로토콜은 요청이 하나 끝나면 정보 삭제. 따라서 로그아웃 시까지 유지시킬 정보가 있을 경우 사용.

로그아웃 할 때까지 유지

application(서버에 올라간 해당 프로젝트 하나): 현재 어플리케이션이 끝날 때(서버에서 내려갈 때)까지 계속 유지

- `daoImpl - select()`
- `editForm.jsp` 파일 생성
- `daoImpl - update()`
- `EditController (servlet)` 파일 생성 => `listcontroller`로 돌아가기 (`sendredirect`)

## - `editForm.jsp`

- `<th></th>`: td와 동일 but 글자 진하게
- el 표기법: `${param명.값}`

EL 표기법

`${}`

- JSP가 실행될 때 즉시 반영된다. (Immediate evaluation)  
- 객체 프로퍼티 값을 꺼낼때 주로 사용

`#{}`

- 시스템에서 필요하다고 판단될 때 그 값을 사용한다. (Deferred evaluation)  
- 사용자 입력값을 객체의 프로퍼티에 담는 용도로 주로 사용

- 기존에 값 가져다 쓰는 방식?

: 변수타입 변수명=(casting type)request.getAttribute("param")

`<%=변수명.getNum()...>`

- el 표기법을 사용하면 장점

1) 여러 스텝을 없애고 바로 값 사용 가능

2) 자바 코드가 없어서 클린 코드로 짤 수 있음

- `${a.num}, ${a.pwd}`

a.pwd는 String이기 때문에.(이미 String이더라도 묶어줘야 됨)

- `getElementById("id명")` => id로 접근(name은 중복 가능하지만 id는 중복 불가)
- get방식 전송: 폼데이터 여러개를 전달할 경우 '&'로 묶어줌

`location.href = "/web0722/guestbook/Edit?content="+str+"&num="+num;`

- location: 객체

location.href: 이동할 페이지 경로

## 2. 스프링 프레임워크 p.53~54

사용자로부터 요청을 받는 서블릿 + 나머지 서블릿 클래스화

- <Command handler 방식> 흐름

1) `guestbook.cmd` 패키지 생성

- 2) [controller가 수행하던 페이지 이동 설정+ 실행 기능 껍데기] 추상 클래스 파일 Command.java 생성  
=> 해당 클래스를 상속받아, 종류별로 약간씩 수정하여 사용
- 3) [Command.java를 상속받은 각 Controller들의 세부명령]해당 추상 클래스 상속받은 ListCommand.java 생성
- 4) [main 컨트롤러] guestbook 패키지에 Controller.java라는 서블릿 생성 p.45
- 5) [main 컨트롤러 하나로 제어할 수 있게 하는 properties 문서]/WEB-INF/cmd.properties 생성  
:: 해당 파일을 읽은 Controller가 name값을 가지고 ListCommand....객체를 생성하여 각자 구현한 execute를 실행.

```

Command.java  ListCommand...  Controller.java  cmd.properties
1 list=guestbook.cmd.ListCommand
2 writeForm=guestbook.cmd.WriteFormCommand
3 write=guestbook.cmd.WriteCommand
4 editForm=guestbook.cmd.EditFormCommand
5 edit=guestbook.cmd.EditCommand
6 del=guestbook.cmd.DelCommand

```

- 6) test

```
localhost:8989/web0722/guestbook/Controller?cmd=list
```

- 7) WriteFormCommand.java 생성  
글작성 버튼 클릭시 form 페이지로의 이동을 담당
- 8) WriteCommand.java 생성  
작성한 글 db에 삽입하고, 다시 첫 페이지로 이동
- 9) form.jsp 파일의 경로 수정  
form의 action 다음 페이지action="/web0722/guestbook/Controller?cmd=write" 로 변경

```

<h3>글 작성</h3>
<form action="/web0722/guestbook/Controller?cmd=write" method="post" name="f">
<tr>

```

- 10) list.jsp 파일의 경로 수정

```

<h3>방명록</h3>
<a href="/web0722/guestbook/Controller?cmd=writeForm">글작성</a><br/>
<table border=1>

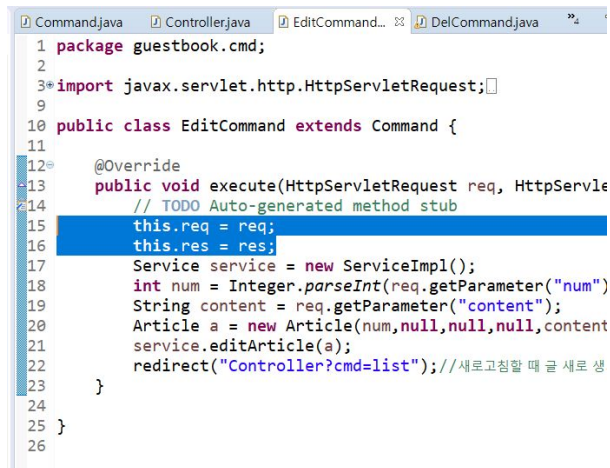
<tr>
  <th>글번호</th><td><a href="/web0722/guestbook/Controller?cmd=editForm&num=<%=a.getNum
</tr>

```

- 11) EditFormCommand.java 생성  
db에서 해당 글번호의 데이터를 가져와서 request에 삽입  
editForm.jsp로 이동
- 12) editForm.jsp파일의 경로 수정  
edit()  
location.href = "/web0722/guestbook/Controller?cmd=edit&content="+str+"&num="+num;  
del()

```
location.href = "/web0722/guestbook/Controller?cmd=del&num="+num;
```

### 13) EditCommand.java 생성 \*req,res 주의



```
1 package guestbook.cmd;
2
3 import javax.servlet.http.HttpServletRequest;
4
5
6
7
8
9
10 public class EditCommand extends Command {
11
12     @Override
13     public void execute(HttpServletRequest req, HttpServletResponse res) {
14         // TODO Auto-generated method stub
15         this.req = req;
16         this.res = res;
17         Service service = new ServiceImpl();
18         int num = Integer.parseInt(req.getParameter("num"));
19         String content = req.getParameter("content");
20         Article a = new Article(num, null, null, null, content);
21         service.editArticle(a);
22         redirect("Controller?cmd=list"); //새로고침할 때 글 새로 생
23     }
24 }
25
26
```

### 14) DelCommand.java 생성 \*req,res 주의