Week4 Review

* 서버- 톰캣 어플리케이션, 아파치 운영, 아차피 톰캣 연동, 톰캣 로드 밸런싱
* 리눅스- 상대경로/절대경로 이해, 시스템 살펴보기, 파일과 디렉토리 조작
* 자바 스프링 – 프로젝트 스프링에 올려 환경변수 설정 및 서버 연결, 화면 확인

1. 서버

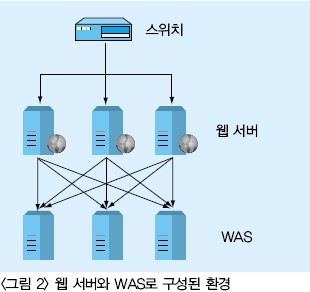
* 아파치와 톰캣의 차이점

웹서버는 정적인 데이터를 처리하는 서버. 이미지나 단순 html(스태틱 파일)을 처리하기 위해 적합.

WAS는 동적인 데이터를 처리하는 서버. DB연결, 데이터 조작 등과 같은 처리에 적합

아파치는 80번 포트, 톰캣은 8080번 포트를 사용

* 웹 서버와 WAS로 구성된 환경: 기능적 분류를 통해 효과적인 분산을 유도. (로드 밸런싱)



1. 리눅스
2. 절대경로, 상대경로

* 절대경로: 최상위 디렉토리 ( / )부터 시작해서 목표 디렉토리까지 가는 경로를 전부 a 에서 하위 디렉토리 d로 가기 # cd /a/b/c/d. 현재 위치 확인 # pwd -> /a/b/c/d

다시 b로 가기 # cd /a/b

* 상대경로: 현재 자신이 있는 위치( . )를 기준으로 이동

b에서 d로 가기 # cd ./c/d # pwd -> /a/b/c/d

d에서 b로 가기 이전 디렉토리는 ( .. ) # cd ../../ # pwd -> /a/b

1. 기본 명령어

* man – 온라인 매뉴얼 페이지
* ls(list segments) – 현재 디렉토리 내용 확인
* pwd(print working directory) – 현재 작업 중인 디렉토리 확인
* cd – 작업할 디렉토리 변경
* mkdir – 새로운 디렉토리 생성 mkdir [폴더이름pictures]
* find/locate – 특정 파일을 찾을 때
* cp – 특정 파일의 사본을 새로운 이름으로 생성하는 명령어

cp cats dogs -> cats 파일의 사본을 dogs란 이름으로 생성

* mv – 파일 이름을 변경하거나 다른 디렉토리로 옮김
* rm 파일을 제거
* more – 파일의 내용을 한 번에 한 화면씩 보여줌
* kill 애플리케이션 실행 중지. 주로 ps 명령어로 실행 중지를 원하는 애플리케이션의 정확한 프로세스 id 확인 후 kill 명령어 사용
* sudo 루트 권한 임시로 부여
* passwd 패스워드 변경
* touch 파일의 용량이 0인 파일을 생성, 날짜 변경
* cat(catenate) 파일의 내용을 화면에 출력하거나 파일을 만듬

cat fname1 fname2 > fname3 : fname1,fname2 를 출력하고 fname3 이라는 파일에 저 장

cat < fname1 : fname1 의 내용을 출력

* redirection 리눅스 스트림의방향을 조정. 화면에 출력되는 결과를 파일로 저장 또는 파일의 데이터를 명령에 입력
* alias 자주 수행하는 명령어들을 쉽게 사용할 수 있도록 설정 (unalias 해지)

alias new = ‘command’ command를 실행하는 새 명령어 new를 만듬

* grep 파일에서 특정 문자열을 찾음.

grep “a” file1 -> file1에서 a가 들어간 행을 화면에 출력

1. 리눅스의 권한(permission) 조정하기(chmod, chown)

* 파일 정보 보기: **“ls -al”** 현재 위치에 있는 파일들을 보여주는 명령어

**drwx------ 1 root root 4096 Jun 21 10:32**

* 파일 타입: “d” 디렉토리, “l” 링크파일, “-“ 일반파일..
* 퍼미션 정보: “r” 읽기 권한, “w” 쓰기, “x” 실행. ‘소유자 – 그룹 – 모든 사용자’
* 링크 수: 해당 파일이 링크된 수 (윈도우의 바로가기 개념)
* 소유자 / 소유 그룹 / 용량 / 생성날짜 / 파일 이름
* 퍼미션 변경하기: chmod [변경될 퍼미션 값] [변경할 파일]
* 퍼미션값: 퍼미션 기호를 숫자로 변환 r = 4, w = 2, x = 1
* 변환한 숫자를 합산 “rwxr-xr-x” 4 + 2 + 1 / 4 + 0 + 1 / 4 + 0 + 1 => 755

“chmod 755 conory.text”

* 디렉토리는 “-R” 옵션 사용시 하위 디렉토리의 모든 디렉토리, 파일 퍼미션 변경

“chmod -R 777 conory : conory 디렉토리의 하위에 위치한 모든 파일, 디렉토리 퍼미션 변경

* 소유자 변경하기: chown [변경할 소유자] [변경할 파일]

1. 사용자 추가 / sudo 권한 부여 / 삭제

* adduser 사용자이름 -> 입력폼 작성 (패스워드 외 엔터눌러서 건너띄기 가능)
* 패스워드가 없는 사용자 추가시 adduser 사용자이름 –disabled-password
* 유저에 sudo 권한 부여하기 **usermod** -aG sudo jeonghwa
* 사용자 삭제 userdel -r jeonghwa
* 유저 리스트 확인: cd /home -> ls -al 유저의 기본 폴더가 ‘/home/유저네임’ 이라는 것을 이용하여 조회 가능

1. Vi 편집기

* 명령 모드: vi 명령어로 vi 시작하면 들어감. 방향키 이용하여 커서 이동. “dd”나 “yy”로한 줄 삭제 및 한 줄 붙여넣기. “x”로 글자 하나 삭제
* 입력 모드: “i” 나 “a”명령을 통해 들어감. 글 작성가능. 명령모드로 돌아오려면 esc
* 마지막 행 모드: 명령모드에서 “:” 입력시 화면 맨 밑단에 입력창 활성화. “:wq” 저장 후 종료. “:q” 저장하지 않고 종료 “:q!” 저장하지않고 강제종료
* 명령어 모음
* 명령모드: u 명령 취소 / dd 현재 줄 잘라내기 (세 줄 3dd) / yy 현재 줄을 버퍼로 복사 / p 아랫줄에 붙여넣기 / $ 맨 뒤로 이동 / G 파일의 맨 끝으로 이동
* 마지막행 모드: :set nu 라인 번호 보기 / 커서 위치 앞으로, 뒤로 문자열 찾기 /문자열, ?문자열

1. 자바 – 스프링
2. 스프링이란

* 자바 언어를 기반으로 다양한 어플리케이션을 제작하기 위한 약속된 프로그래밍 틀: 스프링을 쓰면 톰캣을 이용할 수 있고, 코드의 경량화, 개발 중 테스트가 용이
* **DI**(Dependency Injection): 객체를 외부에서 생성해서 setter()나 contruct()를 이용하여 그것을 사용하는 객체에 넣어줌
* **IoC**(Inversion of Control): IoC컨테이너에서 인터페이스를 통해 부품화된 객체들을 담음
* 스프링이란? 부품을 생성하고 조립하는 라이브러리 집합체
* Bean: 스프링 컨테이너가 생성과, 관계설정, 사용 등을 제어하는 오브젝트

1. 프로젝트 스프링에 올려서 환경변수 설정, 서버 연결, 메인에 접속하여 화면 확인
2. get방식으로 인자값을 받아와서 처리 후 화면에 리턴하기

@Controller

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

@Resource

TestCalculator cal;

@Resource

TestMap map;

@RequestMapping("/cal") // http://localhost:8080/sample/cal?a=123&b=3

**public** String method4(**int** a, **int** b, Model model) {

**int** sum = cal.CalSum(a, b);

**int** sub = cal.CalSub(a, b);

**int** mul = cal.CalMul(a, b);

**int** div = cal.CalDiv(a, b);

System.***out***.println("a와 b의 합은 " + sum);

System.***out***.println("a와 b의 빼기는 " + sub);

System.***out***.println("a와 b의 곱셈은 " + mul);

System.***out***.println("a와 b의 나누기는 " + div);

model.addAttribute("map", map.makeMap(a, b));

**return** "test/test";

}

}

@Service

**public** **class** TestCalculator {

**public** **int** CalSum (**int** a, **int** b) {

**return** a + b;

}

**public** **int** CalSub (**int** a, **int** b) {

**return** a - b;

}

**public** **int** CalMul (**int** a, **int** b) {

**return** a \* b;

}

**public** **int** CalDiv (**int** a, **int** b) {

**return** a / b;

}

}

@Service

**public** **class** TestMap {

**public** Map makeMap(**int** a, **int** b) {

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

map.put("sum", a + b);

map.put("sub", a - b);

map.put("mul", a \* b);

map.put("divi", a / b);

**return** map;

}

}

[test.jsp]

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

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html>

<html>

<head>

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

<title>testView</title>

</head>

<body>

testtest~~~

<br>

<p>합은? <input value = *${map.sum*}></p>

<p>차는? <input value = *${map.sub*}></p>

<p>곱은? <input value = *${map.mul*}></p>

<p>나누기는? <input value = *${map.divi*}></p>

</body>

</html>