Week4 Review

* 서버- 톰캣 어플리케이션, 아파치 운영, 아차피 톰캣 연동, 톰캣 로드 밸런싱
* 리눅스- 상대경로/절대경로 이해, 시스템 살펴보기, 파일과 디렉토리 조작
* 자바 스프링 - MVC패턴 이해하기, DB연동하여 데이터 화면에 출력

1. 서버

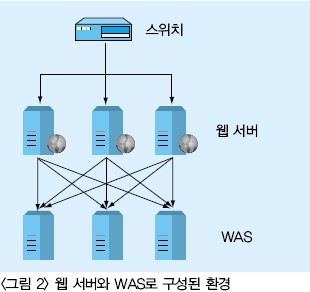
* 아파치와 톰캣의 차이점

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 기술하는 방식. 현재 위치에 상관 없이 똑 같은 명령어를 입력하여 이동

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
* 필요 선행학습: java, JSP&Servlet (MVC모델, model2.? 의 개념)

1. 스프링 프로젝트 만들기

* **DI**(Dependency Injection): 각 계층 사이, 각 클래스 사이에 필요로 하는 의존관계를 컨테이너가 자동으로 연결해주는 것. DI 는 Spring 컨테이너가 IOC를 지원하는 형태로, 클래스 사이의 의존관계를 bean 설정정보를 바탕으로 컨테이너가 자동적으로 연결해줌: 객체를 외부에서 생성해서 setter()나 contruct()를 이용하여 그것을 사용하는 객체에 넣어줌
* **IoC**(Inversion of Control): 프로그램 제어 흐름 구조가 바뀌는 것. IoC는 인스턴스 생성의 제어를 개발자 본인이 아닌, 다른 누군가(bean을 관리해주는 컨테이너)에게 반환해준다는 개념 => 인스턴스의 생명주기 관리를 내가 아닌 컨테이너가 대신 해줌. IoC컨테이너에서 인터페이스를 통해 부품화된 객체들을 담음
* 스프링이란? 부품을 생성하고 조립하는 라이브러리 집합체
* Bean: 스프링 컨테이너가 생성과, 관계설정, 사용 등을 제어하는 오브젝트

1. DI: 스프링을 이용한 객체 생성과 조립

* mainClass 자바 파일에서 직접 객체를 생성하지 않음

public static void main(String[] args) {

String configLocation = “classpath:applicationCTX.xml”;

AbstractApplicationContext ctx = new GenericXmlApplicationContext(configLocation);

// 스프링 컨테이너 생성

MyCalculator myCalculator = ctx.getBean(“myCalculator” 객체의 ID값, MyCalculator.class 객체 타입);

// 스프링 컨테이너에서 컴포넌트를 가져옴

myCalculator.add();

myCalculaotr.sub();

ctx.close(); //컨텍스트를 닫아줌

}

[ctx.xml]

- 다른 파일들과 다르게 경로가 src/main/java 가 아닌 src/main/resources 에 있음

- bean 을 통해서 실행시키는 클래스들을 구현 (Calculator 과 MyCalculator)

- Property 를 통해서 객체안의 필드를 구현 (Calculator calculator, firstnum, secondnum)

<bean id = “calculator” class=”com.javalec.ex.Calculator” />

<bean id =”myCalculator” class=”com.javalec.ex.MyCalculator” >

<property name =”calculator”> //MyCalcuator에 정의된 field: calculator, firstNum, secondNum 을 property로 정의

<ref bean =”calculator”/>

<property name = “firstNum” value =”10”></property>

<property name = “secondNum” value “2”></property>

</bean>

: ~~CTX.xml에서 정의된 것을 가져와서 GenericXmlApplication에서 파싱하여 Context ctx로 저장해주고, 그것을 getBean을 통해 가져오는 데 첫 번째 인자값으로 bean의 unique ID값인 “myCalculator”, 이것의 타입을 명시해줌 MyCalculator.**class**

* 이런 과정을 통해 외부 XML파일에 정의된 객체 myCalculator를 가져와서 정의된 메소드를 사용가능

1. DI 2 – 스프링 프로퍼티 설정, 스프링 컨테이너(IoC컨테이너)의 이해

* 클래스에서 먼저 setter 정의가 다 되어있어야 xml에서 아래와 같이 bean 설정이 가능
* [MyInfo 클래스 파일]

private String name;

private ArrayList <String> hobbys;

private BMICalculator bmiCalculator;

[config.xml 파일] – resources – new – spring bean configuration file

<bean id =”myinfo” class =”com.java.ex.MyInfo” 클래스가 정의된 경로.클래스명>

<property name =”name” value=”홍길동”/> // 기초데이터

<property name =”hobbys” 필드명과 동일하게 줌 > // 리스트타입

<list>

<value>수영</value>

<value>요리</value>

<value>독서</value>

</list>

</property>

<property name=”bmiCalculator” > // 다른 빈 객체 참조

<ref bean=”bmiCalculator” />

</bean>

1. DI 활용: 의존 관계, DI 사용에 따른 장점

* 생성자를 통해 field 초기화: <constructor-arg> 태그 통해 (setter 함수 이용할 땐 <property>태그)

<bean id =”student2” class =”com.java.ex.Student”>

<constructor-arg value =”홍길동”>

<constructor-arg value =”9살”>

<constructor-arg value =”2학년”>

<constructor-arg value =”10번”>

</bean>

* DI 사용의 장점: 규모가 커지고 유지보수 업무가 필요할 때 용이

예) 펜슬 클래스라는 것을 만들어 사용할 경우 **java파일의 수정 없이 스프링 설정 파일만을 수정하여 부품을 생성/조립**할 수 있음

AbstractApplicationContext ctx = new GenericXmlApplicationContext(“classpath:applicationCTX.xml”);

Pencil pencil = ctx.getBean(“pencil”, Pencil.class); // 연필을 사용하는 것은 동일

Pencil.use()

Ctx.use() // 여기까지 코드 변경 필요 없음

* <bean id=”pencil” class=”com.javalec.ex.Pencil4B”/> 4B 굵기로 쓰기
* <bean id=”pencil” class=”com.javalec.ex.Pencil6B” /> 6B 굵기로 쓰기
* <bean id=”pencil” class=”com.javalec.ex.Pencil6BWithEraser” /> 6B 굵기로 지우개와 함께 쓰기

// pencil 에 연결된 클래스(Pencil 인터페이스 내의)만 변경해주면 됨

\*\*인터페이스란?

실제로 구현 코드는 없고, 상수와 메소드 선언만 함. 인터페이스를 사용하고 있는 클래스들은 무조건 오버라이드를 해서 메소드를 만들어야 함: 타입을 통일시킬 수 있고, 강제적으로 오버라이드를 하게 해서 작업을 통일시킬 수 있음. 인터페이스는 “상속(extends)”이라는 표현은 안 쓰고 “구현(implements)”한다고 함

[Pencil 인터페이스]

public interface Pencil {

public void use();

}

[Pencil 인터페이스를 구현하고 있는 Pencil4B 클래스]

Public class Pencil4B implements Pencil {

@Override

Public void use() {

System.out.println(“4b 굵기로 쓰입니다”);

}

}

1. DI설정 방법

* <bean id=”family” class=”com.javalec.ex.Family” c:papaName=”홍아빠” c:mamiName=”홍엄마” p:sisterName=”홍누나”> // “c: ~~” 생성자통한 value 설정. “p:~~” setter함수 통한 value 설정

: 이렇게 쓰려면 <beans xmlns:c=”http:www.springframework.org/schema/c” 부분 추가해야함

1. Get/post방식 사용해서 로그인 정보 받아오기 – 화면 출력

package com.moneta.test.web.controller;

import java.util.Map;

import javax.annotation.Resource;

import org.springframework.stereotype.Controller;

@Controller

public class TestController401 {

@Resource

TestMain testMain;

TestMap map;

@RequestMapping("/main")

public String method1() {

return "test/test1";

}

@RequestMapping("/login")

public String method3(String id, String pw, Model model) { //model: key-value 쌍을 view단으로 보냄

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

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

//문자형 비교할 때는 equals: 상수가 앞에, 변수가 뒤에 들어와야함

model.addAttribute("id", id);

return "test/login";

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

return "test/loginFail";

} else {

return "test/loginFail";

}

}

[Test1.jsp]

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

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html>

<html>

<head>

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

<title>get IdPwd</title>

</head>

<body>

로그인 정보를 입력해주세요.

<form action=*"login"* method=*"post"*>

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

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

<button type=*"submit"*>로그인</button>

</form>

</body>

</html>