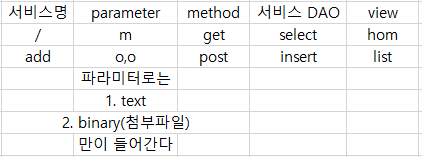
url, parameter(매개변수 같은 것? 서버에게 전송 할 data), method(parameter를 어떤 방법으로 보내줄 것인가? GET 방식, POST 방식(파라미터가 없어도 작동 된다))

get방식: [http://localhost:8080/hom/join?name=’kk’&pwd=’aa](http://localhost:8080/hom/join?name='kk'&pwd='aa)’; 주소에 이름과 비번 같은 정보들이 다 노출 된다.

post방식: 문서에 숨겨놓는 방식. get방식에 비해 은닉화하기 수월하다.

RestAPI : get에는 select, post에는 insert의 의미가 있다. 거기에 delete, put을 추가해서 메소드에 CRUD의미를 담는다. 메소드에 의미를 부여한다?



Client------------------request---------------Server – web server : 정적컨텐츠 서비스 e.g) 첨부파일, 정적문서(html)

- WAS : 동적컨텐츠 e.g) 데이터베이스 작업이 필요한 경우

WAS는 url 안의 context path 보고 정적컨텐츠일지 동적컨텐츠일지 판단하고 넘겨준다.

서비스 연결을 해준다.

e.g)

https:// café or shoping - url보고 판단 cafe-> 스프링 프로젝트

* 서비스 연결 shopping -> 스프링 프로젝트

스프링 레거시는 WAS를 내장하고 있지 않아서 설치해주어야 한다.

스프링 부트는 WAS를 내장하고 있어서 별도의 설치가 필요하지 않다.

스프링 레거시

1. Request Mapping: 요청 분석
2. DB가 필요하면 서비스를 거친다.
3. 서비스를 거쳐 DAO로 간 뒤 DB로 간다.

STS – servers = WAS

- www = 스프링 프로젝트

e.g) <http://localhost:8080/hom/WEB-INF/classes/www/silver/hom/HomeController.java>

Servers / server.xml / 163번째 라인

중 8080/ 까지가 서버의 위치, hom은 context path, hom/ 이후는 Request Mapping에서 정의한(얘가 요청하는게 무엇인지에 대한) 내용, 즉 value 이다.

<Context docBase=*"www"* path=*"/hom"* reloadable=*"true"* source=*"org.eclipse.jst.jee.server:www"*/></Host>

를 보고 확인 할 수 있다.

프로토콜(http://),서버의 주소(localhost), 포트번호(8080), 컨텍스 패스(hom), 밸류(hom 이후 주소)

\* Context path로 어떤 스프링 프로젝트인지 확인하고 뒤의 value를 통해 요청 분석을 한다!

프로토콜: 교육 센터 가는 방법

서버 주소: 휴먼 교육 센터가 있는 빌딩 주소

포트번호: 교육 센터가 있는 층수

컨텍스 패스: 강의실

Value: 강의를 듣는 행위

서비스: 한 명의 고객에게 보여주는 서비스. 고객이 요구하는 서비스를 이해해서 어떠한 DAO를 이용할지 정한다.

DAO: 튜플 하나를 가져오는 기능을 가지고 있다.

DAO는 단일 데이터 접근/갱신만 처리합니다. Service는 여러 DAO를 호출하여 여러번의 데이터 접근/갱신을 하며 그렇게 읽은 데이터에 대한 비즈니스 로직을 수행하고, 그것을 하나의(혹은 여러개의) 트랜잭션으로 묶습니다. 즉, Service가 트랜잭션 단위입니다.

트랜잭션이란 일반적으로 여러 단계 또는 작업의 실행을 필요로 하는 단일 프로그램의 실행을 의미합니다

스프링은 100% 자바 기반. 하지만 클라이언트가 사용했을 웹에서는 자바 인식을 못한다. 그래서 스프링 같은 프로그램은 뷰 템플릿(jsp, thymeleaf)을 이용하여 html 문서로 변환해서 웹이 인식하도록 한다.

cmd -> ping -> ping [www.google.com](http://www.google.com) 뒤의 주소가 살아있는지 확인 가능하다.

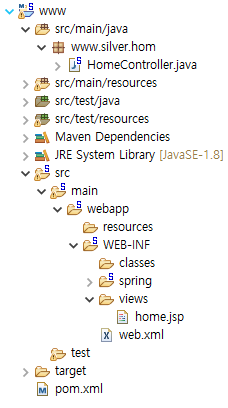
ttl이 뜨면 살아있다는 뜻

Ping www.google.com [142.250.197.132] 32바이트 데이터 사용:

[www.google.com](http://www.google.com) <- 도메인 주소

뒤의 숫자 142.250.197.132 <- 아이피

DNS는 도메인을 아이피로 바꿔주는 기능이 있다.

1. Spring legacy 디렉토리 구조
   1. 자바 소스 위치
   2. 정적 컨텐츠 위치
   3. View 파일 위치
   4. 설정 파일 위치
   5. Maven 설정파일
   6. Mybatis 설정파일 위치

서비스 정의 : 메인화면 보여주세요.

클라이언트 정보

Localhost:8080/hom/

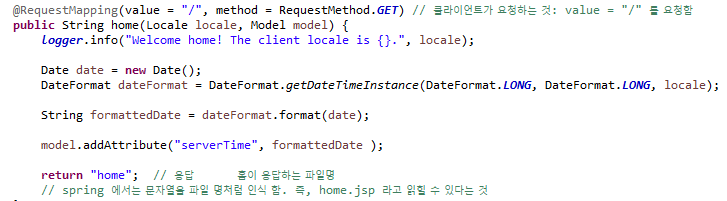
url: /

method: get

parameter: 없음

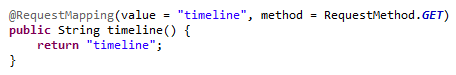
둘은 매칭이 되어야 한다

서버 처리

Mapping: /

데이터베이스 처리 없음

View: home



확인문제1

서비스 정의: 시간표 보여 주세요.

클라이언트 정보 – url: /timeline method: get parameter: 없음

client url: http://localhost:8080/hom/timeline/

서버처리: 데이터베이스 작업 없음 view: timeline

문제> 서버에서 코딩하시오

View가 없어서 말들어 줘야 한다



서버를 재시작하여 테스팅한다.

확인문제2

서비스 정의: 로그인 – 클라이언트가 아이디와 비번을 서버에게 전송하면 서버는 가입자인지 확인한다.

클라이언트 정보 - url: login method: get parameter: userid userpwd

확인문제3

서비스 정의: 지각자만 보기 기능

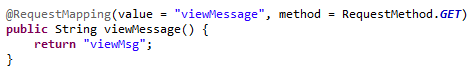
시간표 보여 주세요 서비스에서 <지각자만 보기>라는 메뉴가 있다.

지각자만 보기라는 메뉴를 클릭하면 내일은 일찍 오세요 라고 응답한다.

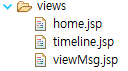
클라이언트 정보 – url: /viewMessage method: get parameter: 없음

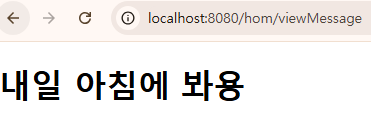
url: <http://localhost:8080/hom/viewMessage>

서버정의: 데이터베이스 작업 없음.. view: viewMsg



뷰가 없으니 만들자





시간표 보여 주세요 서비스에서 <지각자만 보기>라는 메뉴가 있다. 이렇게 정의해서 이대로 코드 수정

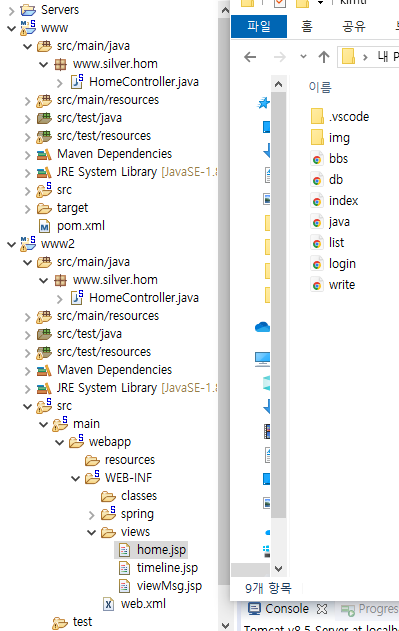
해결) 시간표 view에서 지각자만 보기 메뉴 추가



테스팅은 timeline 요청부터 시작합니다.

A 태그의 request method는 get방식입니다. 주소창에 직접 입력하셔도 get방식입니다.

이번에는 샘플로 제공한 html 파일을 스프링에 이식한다.



404 에러: 컨트롤러로 들어갈 수도 없다. 처리 불가

500 에러. 컨트롤러로 들어갈 수는 있는데 컨트롤러 안에 문제가 있다.

@Controller~ 동기방식을 사용하는 컨트롤러

@RestController~ 비동기방식을 사용하는 컨트롤러

11월 7일

클라이언트에서 url, method, parameter를 전달하고 서버는 처리를 하는 과정입니다.

포인트는,

1. 파라미터를 전달하는 방식인 get, post방식을 이해하고 코드로 작성할 수 있다.

2. 서버는 파라미터를 어떻게 받아(저장한다)내는지 방법을 이해하고 코드로 작성할 수 있다.

첫번 째, 클라이언트에서 파라미터를 get 방식으로 전달하는 법

<a href=”login?uid='kkk'&pw=1234”>

<form action=”login” method=”get”>

<input type=”text” name=”uid”>  
 <input type=”password” name=”pw”>

</form>

두번째, 클라이언트에서 파라미터를 post방식으로 전달하는 법

<a href=”login?uid='kkk'&pw=1234>

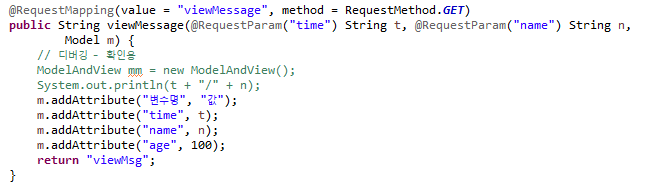
<form action=”login” method=”post”>

<input type=”text” name=”uid”>  
 <input type=”password” name=”pw”>

</form>

컨트롤러에서는 클라이언트가 보낸 파라미터를 받았다.

이제는 컨트롤러가 뷰에게 데이터를 전달하는 예제를 살펴보자.



뷰에서는 전달받은 데이터를 출력합니다. 사용하는 문법은 el 표기법, jstl 문법

JSP template는 el 표기법, jstl 문법을 사용할 수 있다.



컨트롤러가 클라이언트의 파라미터를 받는 방법은 위에서 소개한

@RequestParam() 말고 여러 가지가 더 있다.

아래 주소를 참고하면 된다.

<https://haenny.tistory.com/107>

pom.xml 의  
 <org.springframework-version>4.3.22.RELEASE</org.springframework-version>

<org.aspectj-version>1.8.9</org.aspectj-version>

<org.slf4j-version>1.6.6</org.slf4j-version>  
버전을 바꾸면 알아서 바뀜

Spring주요 용어

DI

컨테이너

AOP

POJO

Mybatis

1. 의존성 설정 <- 디펜던시 추가(라이브러리 빌드)

2. 필요한 Bean 설정(rootContext, servletContext) / Bean이란? 객체다

a. dataSource b. factory c. template

콘솔에 쿼리문이 출력되게 하는 Bean이 있다.

3. client -> 컨트롤러 -> 서비스 -> dao를 간다.

Maven

pom.xml

Spring version

annotation↑

Java version

dependency 종류

Mybatis 퍼시스턴스 프레임워크를 사용하여 Spring 프레임워크와 연동을 한다.

퍼시스턴스 프레임워크는 영속적인 의미를 포함하고 있으며 이 의미는 데이터베이스 작업의 의미로 이해하면 됩니다.

즉, 데이터베이스 작업을 원활히 할 수 있는 프레임워크를 퍼시스턴스 프레임 워크라고 이해하시면 됩니다.

프로젝트 마다 다르지만, 어떤 프로젝트는 mybatis를, 어떤 프로젝트는 JPA를 사용하기도 합니다.

즉 프로젝트마다 다를 수 있다. 그러니.. 퍼시스턴스 프레임워크의 성격을 이해하는 것은 업무의 흐름을 이해하는데 도움이 됩니다.

수업을 듣고 난 후 셀프 성취도 체크 기준

수준 3: 과정을 이해한다.

수준 2: 과정을 이해하고, 스프링 프로젝트의 구조와 연계하여 설명이 가능하다.

수준 1: 과정이해, 구조연계이해, 특징까지 이해하고 문서나 말로써 설명 가능

특: 결합도를 낮춘다는 것의 의미 이해 이해하고 문서나 말로써 설명 가능

Mybatis 작업 과정

1. Maven 에서 Mybatis를 사용하기 위해 필요한 라이브러리 설정.   
   pom.xml 에서 설정하면 자동으로 라이브러리를 다운로드 받아 준다.
2. 컨테이너에게 객체를 등록하기 위한 환경설정을 한다.
3. 서비스 레이어와 dao레이어를 만든다.
4. 데이터베이스 작업을 위한 쿼리문을 정의한다. 이것을 mapper 설정이라고 함

\*\* 순서가 절대적이지 않다.

설정과 개발은 다르다. Denpendency injection이 개발이라면 밑의 servlet/root context는 환경 설정하는 코드이다. 둘은 다르다.

<beans:bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<beans:property name=*"prefix"* value=*"/WEB-INF/views/"* />

<beans:property name=*"suffix"* value=*".jsp"* />

</beans:bean>

Bean 객체

<context:component-scan base-package=*"www.silver.hom"* />

스캔 해서 컨트롤러 옆/위에 @가 안달린 건 스프링이 인식 못하는 클래스가 되어 객체로 만들지

않고 @가 달리면 객체를 만들어서 컨트롤러/서비스 등등의 성격을 부여한다.

<!-- Handles HTTP GET requests for /resources/\*\* by efficiently serving up static resources in the ${webappRoot}/resources directory -->

<resources mapping=*"/resources/\*\*"* location=*"/resources/"* />

url이 <resources mapping=*"/resources/\*\*"* 라고 오면, location=*"/resources/"* />

로 보낸다.

여기는 설정파일이다. 스프링이 구동 될 때 컴파일러가 컨테이너에 객체를 만든다. 개발 코드에서는 필요한 객체를 컨테이너로부터 주소를 주입받는다. 이렇게 컨테이너로 객체를 만들어 놓으면 비지니스 서비스를 개발할 때 접근제어자 같은 것을 생각할 필요 없이 그냥 여기서 주소만 주입받아서 개발한다.

- \*의존성 주입(Dependency Injection, DI)\*: 이는 코드 내에서 객체들 간의 의존 관계를 관리하는 방식이다. 즉, 클래스나 객체가 다른 객체를 필요로 할 때, 이 의존 관계를 코드에서 명시적으로 주입하는 방식. 이를 통해 객체들 간의 결합도를 낮추고, 테스트나 유지보수를 용이하게 만든다. 의존성 주입은 \*\*개발자가 작성하는 코드\*\*에 해당한다.

- \*DI 컨테이너(Dependency Injection Container)\*: 이는 의존성 주입을 실제로 관리하고 제공하는 \*환경 설정 코드\*다. DI 컨테이너는 의존성 그래프를 생성하고, 객체를 생성할 때 필요한 의존 관계를 자동으로 해결해주는 역할을 한다. 예를 들어, Spring이나 Angular 같은 프레임워크에서는 DI 컨테이너를 사용해 객체들을 자동으로 주입해준다.

따라서 \*의존성 주입은 개발자의 코드 내에서 의존 관계를 관리하는 개념\*이고, \*DI 컨테이너는 그런 의존성을 실제로 주입하는 환경 설정을 담당하는 코드\* 라고 할 수 있다. 이 두 개념은 다르지만, 밀접하게 연관되어 있으며, 의존성 주입을 구현하는 데 DI 컨테이너가 중요한 역할을 한다.

클라이언트의 작업:

필요한 src/webapp/web-inf/view/???.jsp html에 가서 원하는 곳에 form태그로 묶고 action method를 준다. 이후, 필요한 인풋에 name을 준다

클라이언트의 작업 끝

VO를 만든다 www.silver.vo 패키지 만들고 안에 VO파일을 만든다.

html의 name이름과 vo의 변수명과 데이터베이스 컬럼은 일치해야 자동으로 매핑 해준다.

컨트롤러가 VO를 클라이언트로부터 받아 서비스에 조인시킨다.

서비스단에서 서비스 인터페이스를 정의. 컨트롤러가 조인 작업을 요청한다. 이 때, 매개변수로 객체의 주소를 전달

서비스impl에서는 VO의 객체를 매개변수를 받았기 때문에 이를 DAO에 넘긴다. 해당 클래스가 객체로 만들어져야만 전달이 가능하기에 @Service어노테이션을 단다.

\*\*\*스프링은 하나의 클래스는 하나의 인터페이스만 가질 수 있다.

이후 DAO는 DB로 crud를 하는 것과 같은 작업을 하게 된다.

1. 클라이언트가 요청한 value / e.g) localhost:8080/hom/board

2. 404가 뜨는 이유

컨트롤러가 value를 인식할 수 없음

즉, 컨트롤러에 정의가 되어 있지 않다.

이 가능성이 제일 높다.

코드를 작성하다가 404를 만나면 컨트롤러를 체크하시면 됩니다.

실제 데이터에 대한 정보를 갖고 있는 것이 .metadata이다.

파일 type의 폼의 name이 같다면 둘이 배열로 나열되어 올라간다.

encType=*"multipart/form-data"* <-넣어아 첨부파일이 서버로 전송 됨

주제: 파일 업로드

필요한 spring 라이브러리 : multipart

클라이언트

- input type=”file” name = “file”

- encType 지정

Spring

- 라이브러리 설정 > pom.xml

<!-- 파일업로드 라이브러리 -->

<dependency>

<groupId>commons-fileupload</groupId>

<artifactId>commons-fileupload</artifactId>

<version>1.3.1</version>

</dependency>

- 컨트롤러에서 파일이 정상으로 업로드 되는지 확인

이 때, servlet-content.xml 파일에 빈 설정이 되어야 정상적으로 첨부파일을 받을 수 있다.

<beans:bean class="org.springframework.web.multipart.commons.CommonsMultipartResolver" id ="multipartResolver">

<beans:property name="maxUploadSize" value="10485760"></beans:property> <!-- 10메가 제한 설정-->

</beans:bean>

<beans:bean class= "java.lang.String" id="uploadPath">

<beans:constructor-arg value="c:\\tmp"></beans:constructor-arg>

</beans:bean>

- 업로드 된 파일의 이름을 바꾸고, 파일을 저장하는 작업

첫 번째 작업은 이름을 바꾸고 파일을 저장하는 코드 작업

이 코드는 공유파일 FileDatUtil로 배포하였습니다.

- 이제는 컨트롤러에서 클라이언트로 받은 file을 fileDataUtil 객체에게 매개변수로 전달하고, 파일명만 배열 타입으로 리턴 받는다. 이렇게 되면, 지정된 위치에 첨부파일을 저장하게 되고 이제는 리턴받는 변경된 파일명을 디비에 저장하는 작업을 하면 된다.

- 컨트롤러는 FileDataUtil 객체가 필요하다.

그래서 이 객체를 컨테이너로부터 주입받겠다.

컨테이너로부터 주입받으려면 FileDataUtil 객체가 컨테이너에 있어야 하는데, 이를 위해서 @Controller를 작성하였고 Scan설정도 해 주었다.

- 리턴 받은 파일명을 디비에 저장하기 위해서 기존은 boardVO에 변수를 추가하고 변수에 리턴 받은 파일명을 저장한다.

**ALTER** **TABLE** kboard **ADD** ***num*** int;

**CREATE** **SEQUENCE** kboard\_seq;

**ALTER** **TABLE** kboard **ADD** **PRIMARY** **key**(num);

유니크 속성 추가