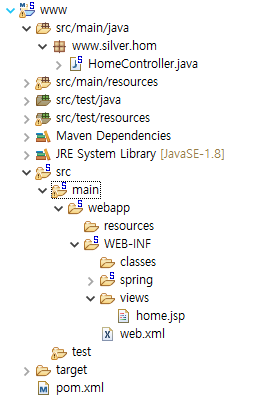
1. Spring Legacy 디렉토리 구조(maven build)



* 1. 자바 소스 위치
  2. 정적 컨텐츠 위치
  3. View 파일 위치
  4. 설정파일 위치
  5. Maven 설정파일
  6. Mybatis 설정파일 위치

서비스 정의 : 메인화면 보여주세요

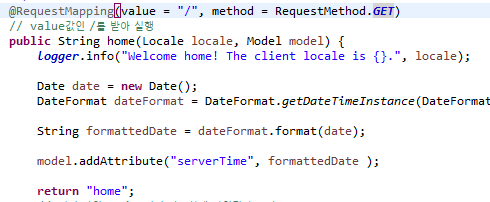
클라이언트 정보

URL : /

Method : get

매칭시켜준다.

Parameter : 없음

서버 처리

Mapping : /

데이터베이스 처리 없음

View : home

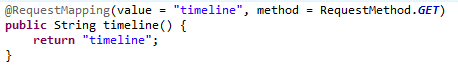
확인문제

서비스 정의 : 시간표 보여 주세요 .

클라이언트 정보

URL : timeline / Method : get / Parameter : 없음 / 서버처리 : 데이터베이스 작업 필요없음 / View : timeline

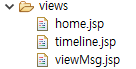
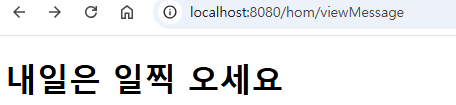
문제 > 서버에서 코딩하시오.



클라이언트 URL : <http://localhost:8080/hom/timeline>

View가 없어서 만들어 줘야함





확인문제

서비스 정의 : 지각자만 보기 가능

시간표 보여주세요 서비스에서 <지각자만 보기> 라는 메뉴가 있다.

지각자만 보기 라는 메뉴를 클릭하면 내일은 일찍오세요 라고 응답한다.

클라이언트 정의 : URL : /viewMessage / method : get / parameter : 없음

서버 정의 : 데이터베이스 조작 필요 없음 / View : viewMsg

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

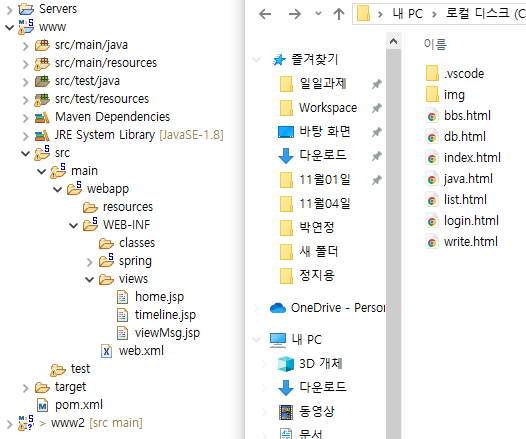
테스팅

시간표 보여주세요 서비스에서 <지각자만 보기> 라는 메뉴가 있다 이렇게 정의해서 수정

시간표 view에서 지각자만 보기 메뉴를 추가시키겠다.

a 태그의 request method는 get방식입니다. 주소창에 직접 입력하셔도 get방식입니다.

이번에는 샘플로 제공한 html 파일을 스프링에 이식한다.



Img 파일은 정적 콘텐츠이므로 resources에 저장

Html 파일은 사용자에게 보여질 화면인 view에 해당하므로 views에 저장한다.

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

\* 포인트

파라미터를 전달하는 방식인 get, post 방식을 이해하고 코드로 작성할 수 있다.

서버는 파라미터를 어떻게 받아내는지(저장하는지) 방법을 이해하고 코드로 작성할 수 있다.

첫번째, 클라이언트에서 파라미터를 get 방식으로 전달하는 법

<a href=”/login?name=kk&pw=1234”>

<form action=”login” method=”get”>

<input type=”text” name=”uid”>

<input type=”text” name=”pw”>

</form>

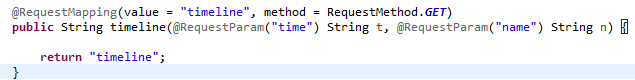
두번째, 클라이언트에서 파라미터를 post방식으로 전달하는 법

<form action=”login” method=”post”>

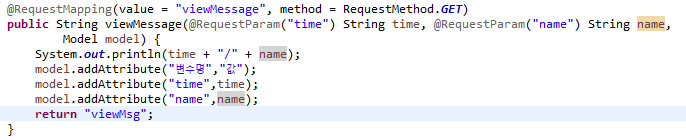
<input type=”text” name=”uid”>

<input type=”text” name=”pw”>

</form>

컨트롤러에서는 클라이언트가 보낸 파라미터를 받았다.

이제는 컨트롤러가 뷰에게 데이터를 전달하는 예제를 살펴보자



모델을 사용해서 클라이언트에게 파라미터 값을 넘겨준다.

뷰에서는 전달받은 데이터를 출력합니다. 사용하는 문법은 el표기법, jstl문법

JSP template은 el 표기법, jstl문법을 사용할 수 있다.

컨트롤러가 클라이언트의 파라미터를 받는 방법은 위에서 소개한

@RequersParam()말고 여러가지가 더 있다.

아래 주소를 참고하면 된다.

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

1. 의존성 설정 > 디펜던시 추가해라(라이브러리 빌드해라)

Maven에서는 pom.xml에서 했다.

pom.xml에서는 Spring 버전, Java 버전, Dependency종류를 확인

2. 필요한 bean 설정 (root-context, servlet-context에서)

1) dataSource

2) factory

3) templete

(콘솔에 쿼리문이 출력되게 하는 bean도 있음)

3. 작업순서 Client > controller > service > DAO

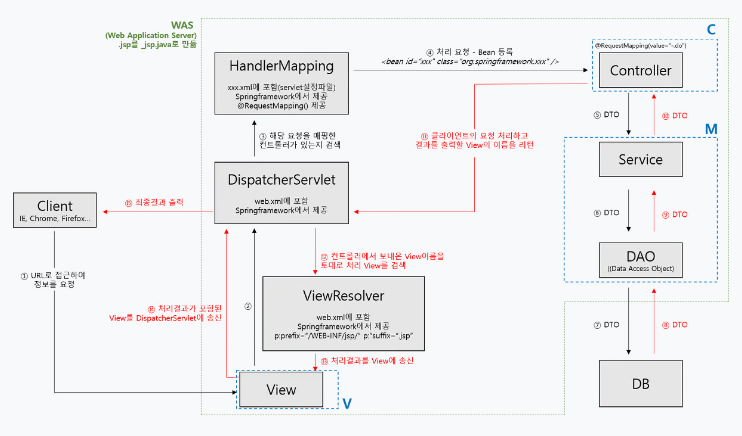
객체를 만들고 싶은 클래스 위에 어노테이션을 붙이면 되는데 내가 개인적으로 만들고 싶은게 있으면 Component를 하면 된다.

Inject <> Autowired 둘 다 주소를 주입받는 어노테이션이지만 주입하는 방법이 다르다.

컨테이너는 객체가 아니라 객체를 저장하는 공간이다. 내가 객체를 컨테이너에 등록하냐 마냐 판단해서 한다. root.xml에 bean 설정을 하거나 어노테이션을 붙이거나 선택해서 한다. 객체가 만들어졌다는 것은 주소값이 할당되었다는 뜻 이고 그 이후에 컨테이너에 등록이 되는 것 이다.

https://velog.io/@chlgustjr/SpringBoot-Controller-Service-Repository-%EC%9D%98%EC%A1%B4%EC%84%B1-%EC%A3%BC%EC%9E%85

https://velog.io/@dyunge\_100/Spring-%EC%8A%A4%ED%94%84%EB%A7%81-%ED%94%84%EB%A0%88%EC%9E%84%EC%9B%8C%ED%81%ACSpring-Framework%EC%9D%98-%ED%8A%B9%EC%A7%95%EA%B3%BC-%EA%B5%AC%EC%A1%B0



IoC(제어의 역전)

프로그램의 제어 흐름을 직접 제어하는 것이 아니라 외부에서 관리하는 것을 제어의 역전이라함.2024-11-13

게시판 페이징 관련 정리하기

2024-11-14

이미지 업로드 정리하기

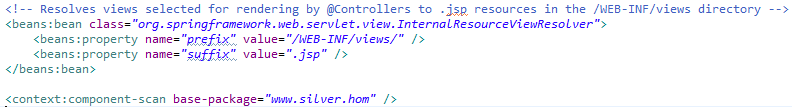
@ResponseBody

리턴 타입이 view가 아니다.

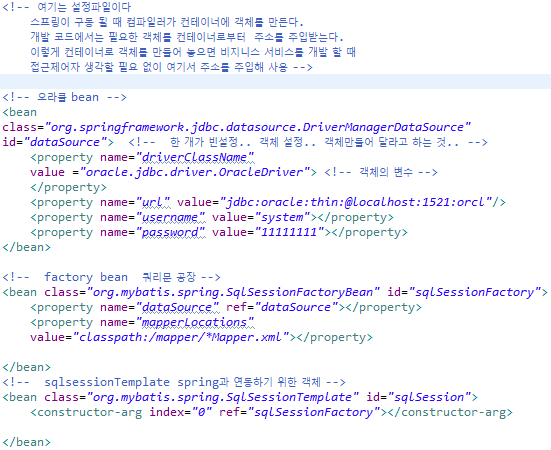
뷰 리졸브에 영향을 받지 않겠다.

www/pom.xml

/www/src/main/webapp/WEB-INF/spring/appServlet/servlet-context.xml



/www/src/main/webapp/WEB-INF/spring/root-context.xml



1. dataSource
   1. 데이터베이스와 연동하기 위한 객체
2. sqlSessionFactory
   1. 쿼리문을 저장한 객체
3. sqlSession
   1. 서비스와 연동하기 위한 객체

정리질문

1. 클아이언트가 request를 할 때 공부해야 하는 것 3가지
2. URL의 구성하는 각 부분을 설명하시오
3. 스프링에서 @Controller의 의미
4. 스프링에서 RequestMapping의 의미
5. 스프링 컨트롤러에서 return 다음에 오는 문자열의 의미
6. Maven Spring의 디렉토리 구조
7. 이미지나 정적컨텐츠의 위치
8. 스프링에서 한글을 설정하는 방법 3가지
9. 컨텍스트패스의 의미
10. 클라이언트에서 get 방식으로 request를 발생하는 방법 2가지

---- 추가하면 좋은 이론들

DNS, IP, PORT

REST API

Web server & WAS

View Template

1. 클라이언트가 request할 때 공부해야하는 것 3가지
   1. URL

URL은 네트워크 상에서 파일이 위치한 정보를 나타낸다.(자세한 설명은 2에서)

* 1. Method

Method는 클라이언트가 웹 서버에게 어떤 동작을 원하는지를 나타내는 방법이다.

GET, POST, HEAD, PUT, DELETE, PATCH, OPTIONS, TRACE, CONNECT 중에서

* + 1. GET

주로 서버에서 정보를 조회할 때 사용한다.(CRUE - R)

GET 방식은 정보가 URL에 보여진다.

장점 – 간편성, 캐시 사용을 통한 이전 결과 재사용성, 가시성

단점 – 보안 취약성, 길이 제한

* + 1. POST

주로 서버에서 리소스를 추가할 때 사용한다.(CRUD - C)

POST 방식은 정보가 HTML 내용 안에 포함하여 전송한다.

장점 – 보안성, 데이터 길이 제한 없음, 상태 변경 가능

단점 - GET방식보다 느림, 캐시 미사용, 복잡성

* + 1. PUT

특정 URL에 대응하는 리소스의 전체 내용을 갱신하는 데 사용한다.(CRUD – U)

* + 1. DELETE

특정 리소스를 제거하는 데 사용한다.(CRUD – D)

* 1. Parameter

파라미터는 웹 사이트에 접속했을 때, 특정 값이 미리 입력 되어있게끔 만드는 기능

파라미터는 웹사이트 주소(URL) 뒤에 ?를 붙여 추가하며,

[파라미터 변수] = [파라미터 값]으로 선언한다.

클라이언트가 REQUEST할 때 파라미터가 있을 수도 있고, 없을 수도 있다.

파라미터는 결제방식에 카드느낌(?)

1. URL의 구성하는 각 부분을 설명하시오

[http://www.example.com/example](https://www.example.com/example)

1. Scheme(스킴) >http : 어떤 방식으로 서버와 통신하고 싶은지를 명시

인터넷 상에서 데이터를 주고 받기 위한 \*클라이언트-서버 모델을 따르는 프로토콜

\*클라이언트-서버 모델?

클라이언트 >서비스를 요청하는 측

서버 >서비스를 제공하는 측,

1. Host(호스트) >[www.example.com](http://www.example.com) : 웹사이트를 운영하는 서버 또는 컴퓨터의 주소

DNS서비스를 통해 사람들이 읽을 수 있는 주소로 IP주소를 변환해 컴퓨터에게 전송

1. Path(패스) >/example : 호스트를 통해 서버에게 어떤 요청을 할 것인지 명시

<http://localhost:8080/hom/word>

http > 프로토콜 / localhost > 서버의 주소 /

8080 > 포트번호 / hom > \*컨텍스트 패스 / word > 벨류

\*컨텍스트 패스?

여러개의 프로젝트를 생성해도 하나의 WAS로 처리하기 때문에 프로젝트를 구분해야한다.

서로간의 프로젝트를 구분하기 위해 WAS에서 server.xml 파일에 설정을 한다.

1. 스프링에서 @Controller의 의미
   1. 요청을 받아준다. 이때 파라미터가 있다면 받아 준다.
   2. 비지니스 로직을 수행한다.

비지니스 로직이란 데이터베이스 작업을 의미한다.

즉 비지니스 로직이 필요하면 서비스 레이어에 작업을 시킨다.

* 1. 응답을 해준다.

동기방식에서는 view를 지정한다.

비동기방식에서는 문자열, 또는 숫자, 객체로 전송한다.

1. 스프링에서 RequestMapping의 의미
   1. value와 메서드를 지정하여 클라이언트의 요청을 처리해주는 어노테이션이다.

\* 어노테이션이란? @로 시작하며 컴파일러에게 특수목적이 있음을 알려준다.

스프링에서는 어노테이션으로 스프링에서 제공하는 기능을 부여한다.

1. 스프링 컨트롤러에서 return 다음에 오는 문자열의 의미
   1. view 이름
2. Maven Spring의 디렉토리 구조
   1. pom.xml < 버전 설정, 라이브러리 관리 등
3. 이미지나 정적컨텐츠의 위치
   1. /프로젝트명/src/main/webapp/resources
4. 스프링에서 한글을 설정하는 방법 3가지
   1. IDE 환경 설정에서 UTF-8 설정
   2. 프로젝트 내 web.xml 환경변수 설정
   3. JSP 메타태그 설정
5. 컨텍스트패스의 의미
   1. WAS 서버가 여러개의 프로젝트를 구분하기 위해 설정하는 경로
6. 클라이언트에서 get 방식으로 request를 발생하는 방법 2가지
   1. 웹 브라우저 URL에 표기하여 요청하는 방식
   2. 앵커태그를 사용해 요청하는 방식

1. 스프링 레이어의 의미

계층 구조 (Layerd Architecture)란 코드를 모듈화하고 재사용성을 높여, 애플리케이션을 쉽게 확장하고 유지보수 하기 위해 사용한다. 이를 통해 애플리케이션 전체의 유지보수성이 향상된다.

스프링에서 레이어는 Controller - Service - Repository로 나뉜다.

\*@Controller, @Service, @Repository어노테이션을 선언해 스프링 컨테이너에 Bean 등록한 상태

**Controller**

컨트롤러는 클라이언트의 요청을 받는다.

서비스 객체는 스프링 컨테이너에 있는 서비스 Bean을 주입받아 서비스와 연결컨트롤러에서 서비스를 호출하고 의존성 주입(@Inject & @Autowired) 을 통해

**Service**

서비스는 컨트롤러부터 받은 데이터를 이용해 비즈니스 로직을 처리한다.

서비스또한 DAO를 호출하고 의존성 주입을 통해 DAO객체는 Bean을 주입받아 서비스와 연결

**Repository**

리포지토리는 비즈니스로직에서 DB조작을 분리하기 위해 사용한다.

SqlSession객체를 호출하고 의존성 주입을 통해 필요한 데이터베이스 작업을 수행

2. 게시글 쓰기를 클라이언트, 컨트롤러, 서비스, 리포지토리, mapper, DB

다시 DB, 리포지토리, 서비스, 컨트롤러, 뷰 순으로 정리

클라이언트가 요청한 벨류값을 확인하고 게시글에 입력한 값을 서비스로 넘겨준다.

서비스에서 필요한 처리작업 진행 후 DB작업을 수행하기 위해 DAO로 전달한다.

리포지토리에서 sqlsession을 사용하여 mapper.xml에 정의해놓은 데이터베이스 작업을 수행한다.

작업이 수행되면 클라이언트요청을 완료하고 컨트롤러에서 게시판 뷰를 리턴한다.

3. 게시글 전체보기를 클라이언트, 컨트롤러, 서비스, 리포지토리, mapper, DB

다시 DB, 리포지토리, 서비스, 컨트롤러, 뷰 순으로 정리

클라이언트가 요청한 벨류값을 확인하고 리턴값이 있는 전체보기 요청을 서비스로 넘겨준다.

서비스에서 필요한 처리작업 진행 후 DB작업을 수행하기 위해 DAO로 전달한다(리턴 값 O).

리포지토리에서 sqlsession을 사용하여 mapper.xml에 정의해놓은 데이터베이스 작업을 수행한다.

작업이 수행되면 리턴값을 받아와 model객체에 저장하고 뷰로 데이터를 전달한다.

jsp파일에서 JSTL문법을 사용해 반복문을 만들어 게시글을 작성한다.

클라이언트요청을 완료하고 컨트롤러에서 게시판 뷰를 리턴한다.

4. 게시글 삭제를 클라이언트, 컨트롤러, 서비스, 리포지토리, mapper, DB

다시 DB, 리포지토리, 서비스, 컨트롤러, 뷰 순으로 정리

클라이언트가 요청한 벨류값을 확인하고 게시글을 삭제할 값을 서비스로 넘겨준다.

서비스에서 필요한 처리작업 진행 후 DB작업을 수행하기 위해 삭제할 값을 DAO로 전달한다.

리포지토리에서 sqlsession을 사용하여 mapper.xml에 정의해놓은 데이터베이스 작업을 수행한다.

작업이 수행되면 클라이언트요청을 완료하고 redirect:를 통해 컨트롤러에서 뷰를 리턴한다.