


백엔드 연동
기름 연동



백엔드 연동

인텔리제이 > 새 프로젝트 > 제네레이터: Spring Boot > 3.2.6 버전

이름: springbootdemo

그룹: com. bit
(회사명)

> create

✓ Spring Boot Devtools

✓ Lombok

✓ Spring Configuration Processor

✓ Spring Web

✓ Thymeleaf

✓ MyBatis Framework

✓ Oracle Driver

- 설정 > Auto Import > ☒ 모호하지 않은 import문 즉시 추가
☒ import문 즉시 추가

onclick 이벤트가 발생하면서

① 회원가입 버튼을 누르면 url을 /join으로 변경

<input type="button" value="회원가입" onclick="location.href='/join'" >

루트path

*
Java → ① RestController
: 해당 클래스를 http 요청을 받아서
처리할 수 있는 클래스로 만들어줌
+ ① Post Mapping ("/join")

브라우저는 url을
/join으로 변경하고
해당 url로 HTTP
GET 요청을 서버에
보냄. 서버는 /join에
들어온 GET 요청을 처리하기
위해 매핑된 메소드 실행

(주소에 루트path가 추가됨...)

url을 매핑해놓은 것

↓
Java에 해당 url로
매핑된 메소드를 만들 수 있게 됨..

* Q. ① Controller와의
차이점 이해 못함...

↓
② "/join" url로 매핑된
메소드로 이동

③ join 메소드가
join.html을 반환

④ join 페이지로 이동

main > Java 에 클래스 생성

↓
① Get Mapping ("/join")

ModelAndView 객체를
리턴 타입으로 갖는 join 함수를 생성,
ViewName 을 join으로 바꾸고
객체 다시 반환

↓
/join 페이지로 이동!

⇒ 백엔드에 url로 매핑된 메소드를 만들어 놓고 그 메소드가 실행되면서

html에서 form 태그의 action문

프론트에서 받아온 데이터를 처리

㉞ PostMapping ("/login")

```
public void login() { }
```

㉟ PostMapping ("/join")

```
public ModelAndView join() { }
```

리턴 타입

매개변수 x join 화면으로만 이동해주는 기능

ModelAndView 객체 생성

```
ModelAndView mav = new ModelAndView();
```

```
mav.setViewName("join"); → 뷰 이름 "join"으로
```

```
return mav; → 뷰 이름이 반환됨 → join.html로 이동
```

Q. 여기서 ModelAndView 객체는 이미 만들어져 있는 객체?

A. 정답. ModelAndView 객체는 스프링 프레임워크에서 제공하는 클래스이다

이를 이용하여 컨트롤러 메소드에서 모델 데이터와 뷰 이름을 반환!

뷰(HTML 페이지)에서 사용할 수 있는 데이터
login.html 같은

① 로그인 페이지에서 회원가입 버튼 클릭



② GET 요청으로 매핑된 join 함수 실행



join 메소드의 반환값 실행



회원가입 페이지로 이동



가입할 아이디, 비밀번호 입력 후
회원가입 버튼 클릭



Post 방식으로 매핑된 join 함수 실행

* 같은 함수를 다른 방식으로 생성 가능

@GetMapping, @PostMapping으로

onclick = "location.href = '/join'"



@GetMapping("/join")

public ModelAndView join() {

객체 생성

map.setViewName("join")

return map



join.html 실행

<form action="/join" method="post">

⋮

<input type="submit" value="가입">



@PostMapping("/join")

public M-A-V join() {

파라미터가 너무 많아지면
클래스로 만들어 입력

@Setter (lombok 라이브러리)

public class MemberDto {

private long id;

" String username;

" " password;

MVC 패턴?

: 웹개발에서 자주 사용하는 설계 패턴으로 아래와 같이 나뉜다.

- Model : DB와 연동되는 service, DAO, Mapper
- View : 화면단 (html, jsp)
- Controller : 화면에서 데이터를 받거나 DB의 데이터를 화면으로 넘겨주는 클래스

이와 같이 설계구역을 나누어 개발을 함.

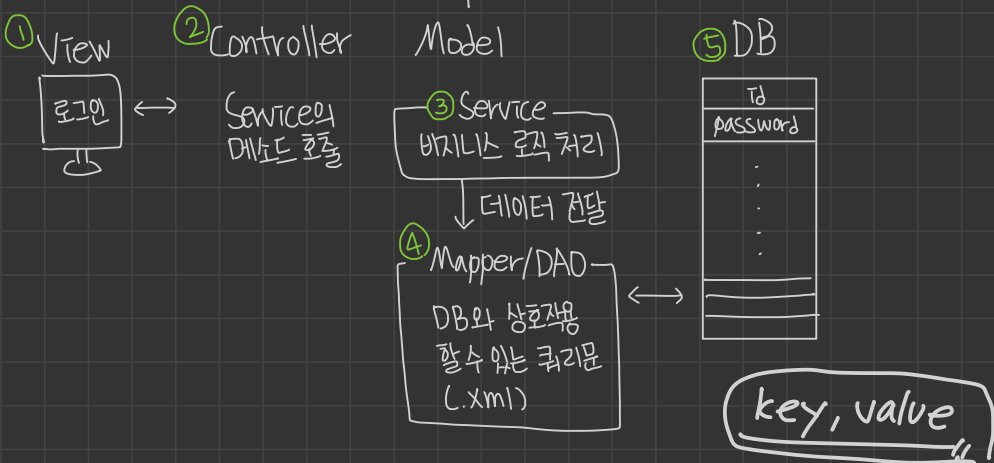
↳ 각각의 계층이 명확하게 분리됨으로서 코드의 유지보수성과 가독성 향상..

또 다른 패턴으로는 MVP, MVVM, Flux/Redux 등이 있음

<웹 페이지의 데이터 흐름>

DB에 대한 CRUD기능 제공

보통 Map형태 (key-value)로 데이터가 저장됨



- Service 계층은 컨트롤러와 데이터 액세스 계층(DAO/Mapper) 사이의 중재자 역할
비즈니스 로직 처리 (비밀번호 암호화 로직 등을 처리)

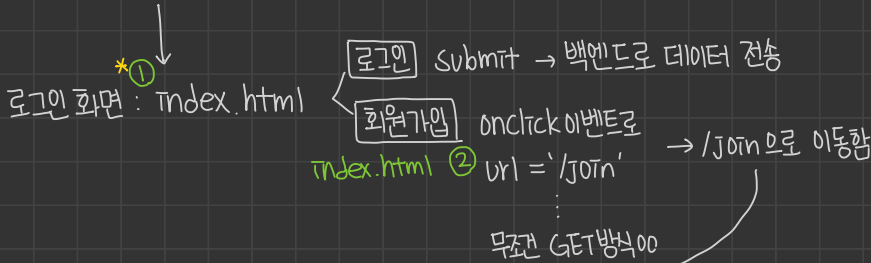
- Mapper : Mybatis 연동으로 인해 쿼리문을 xml 형태로 가지고 있음

메소드 이름으로 된 ID값으로 쿼리문 생성 가능

↳ CRUD 기능 실행 (INSERT, UPDATE, DELETE)

local host : 8090

* 컴파일러가 파일이름이 index인것을 첫 페이지에 띄워줌



MemberControl.java ③

@GetMapping에서 setViewName("join") 실행

↓ ④ join.html

(form 태그에 action을 join으로 매핑시켜놓음)

회원가입 화면 : join.html → id와 pw 입력 후 회원가입 submit (post방식)

⑤ join.html

↓

MemberController.java ⑥ Controller의 @PostMapping (/join) 실행
(join 메소드 실행)

↓

⑦ username과 password가 담긴 MemberDto 객체에
사용자가 입력한 값을 담아주게 됨

↓

⑧ MemberDto 객체를 Service Impl이 있는
join 메소드에 전달 (8-1: Service 인터페이스로 갔다가)

↓

MemberServiceImpl.java ⑨ 해당 클래스의 join 메소드에서

보통 암호화 작업 실행

⑩ 이서 받은 MemberDto 객체를 매개변수로 받고
Mapper이 있는 join 메소드로 전달

↓

! 현업에서도
Service는 인터페이스로
구현부는 Impl 파일로
만들어서 코드를 짜요..

장호님

MemberMapper.java ⑩ 매개변수로 받고 JOIN 함수 실행

여기서는 이어주는 역할만!
이렇게 영역별로 역할을
나눠줘야 나중에 유지보수에 편함,,

+ @Mapper 어노테이션을 통해 xml 파일과 매핑
(매개변수로 MemberDto 객체를 xml로 전달)

MemberMapper.xml ⑪ <mapper namespace> 태그로 mapper의 위치를 지정하면
mapper에서 받은 값을 쿼리문을 통해 DB에 저장/수정

*html에서 DB의 데이터를 꺼낼때, Controller에 지정된 key값으로 꺼내주어야 한다.