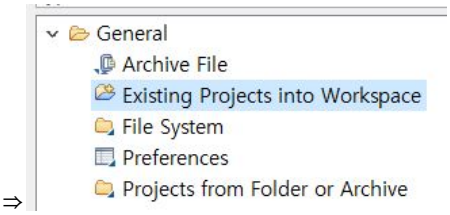
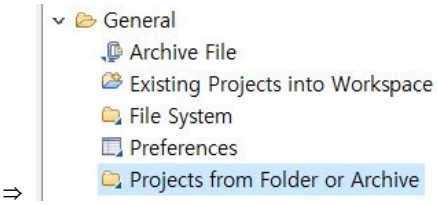
- import
- 이미 eclipse 환경에 적용된 파일

⇒ 

- Spring Boot의 zip파일

⇒ 

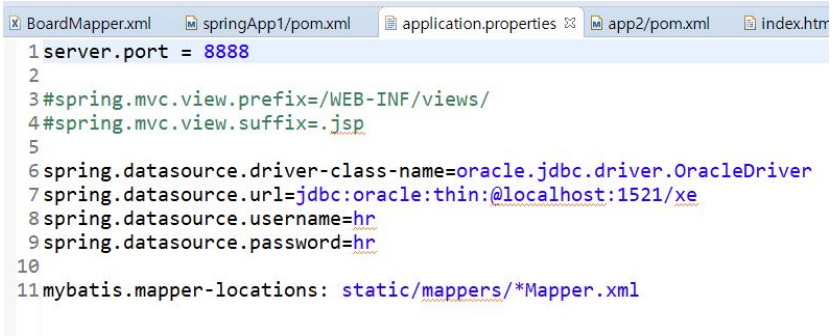
<복습: mybatis를 이용>

resulttype
- 기본 타입 or 클래스 타입 사용할 때

resultmap 사용하는 경우
- 기본 타입이 아닌 클래스 타입 사용할 때 쓰는 것이 아니라
- join했을 때나
- 컬럼이름과 dto맴버 이름이 다를 때 매핑해주기 위함

1. mybatis (-ing)
   스프링 부트 내에는 자체 서버 존재(내장 톰캣)
   자체 서버를 사용하겠다는 설정.

- application.properties



- pom.xml (jsp, jstl, autoreload 추가)
  *auto: 개발하는 동안에는 수정잦기 때문에 true로, 개발 완료후에는 false로 바꾸기
  : 스프링 부트 내장 톰캣 사용할 경우, JSP 사용하기 위해서

BoardMapper.xml | springApp1/pom.xml | application.properties | *app2/pom.xml | index.

springApp1/src/main/resources/static/mappers/BoardMapper.xml

```
        </dependency>
        <dependency>
        <groupId>org.mybatis.spring.boot</groupId>
        <artifactId>mybatis-spring-boot-starter</artifactId>
        <version>2.1.3</version>
        </dependency>

        <dependency>
        <groupId>org.apache.tomcat.embed</groupId>
        <artifactId>tomcat-embed-jasper</artifactId>
        </dependency>
        <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <optional>true</optional>
        </dependency>
```

- 홈

springApp1/... | application... | app2/pom.xml | index.html

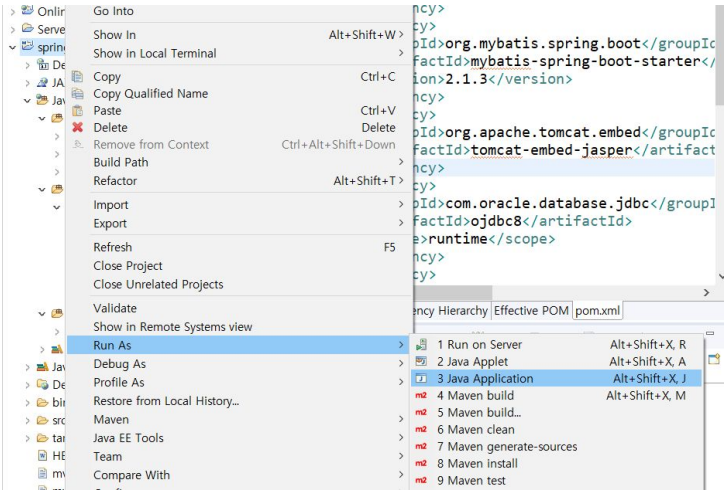
```
<!DOCTYPE html>
<html>
<head>
<meta charset="EUC-KR">
<title>Insert title here</title>
</head>
<body>
<H3>hello spring</H3>
</body>
</html>
```

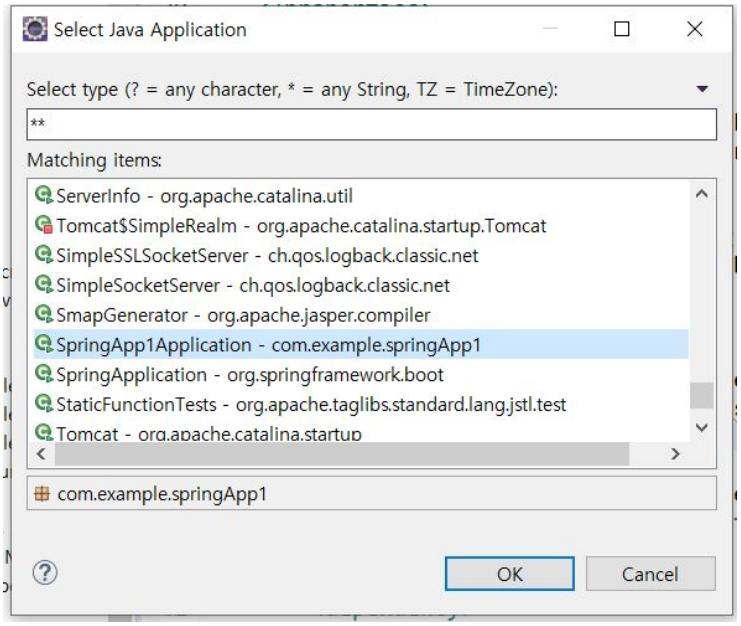
- 컨트롤러

app2/pom.xml | index.html | HomeControll...

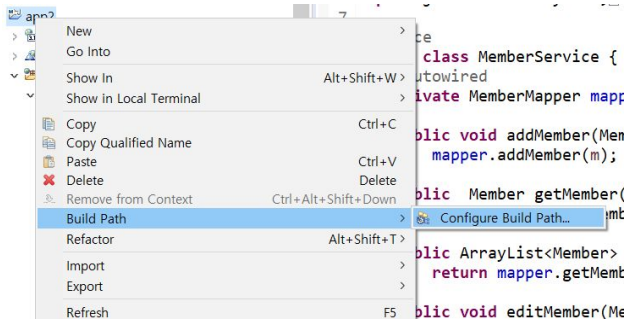
```
package com.example.app2;

import org.springframework.stereotype.C
import org.springframework.web.bind.ann

@Controller
public class HomeController {
    @GetMapping("/")
    public String home(){
        return "/index.html";
    }
}
```

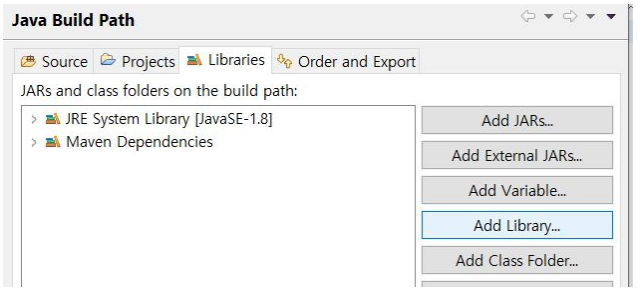
- RUN

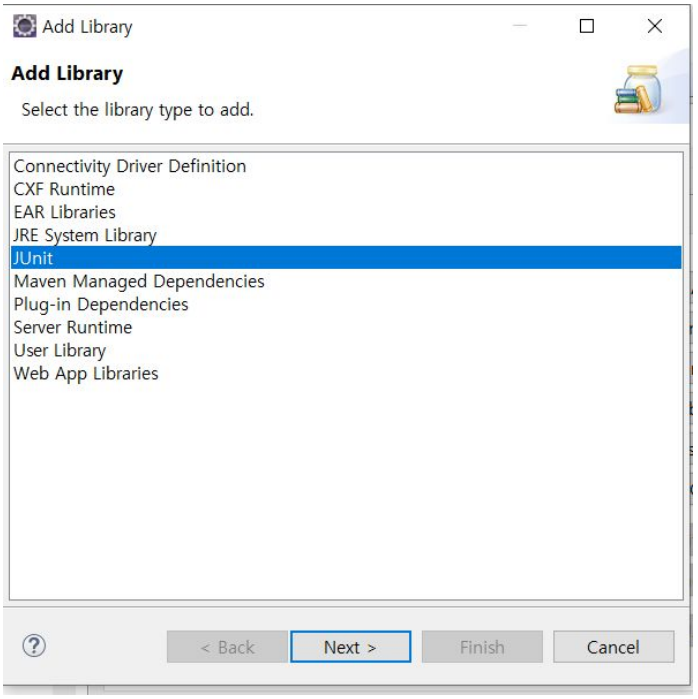


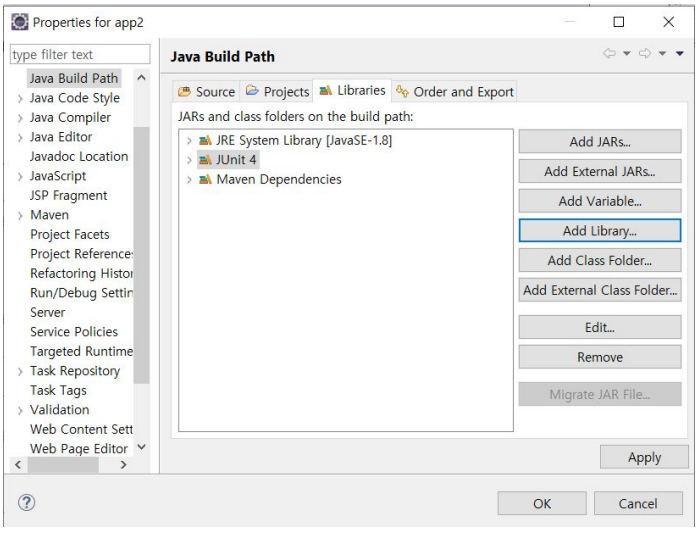


- 페이지 접속
  localhost:8888
- 복붙
  - mappers 폴더(내부 내용 수정)
- 단위 테스트: jUnit 사용 (pom.xml에서 확인 가능)
  - 라이브러리 추가

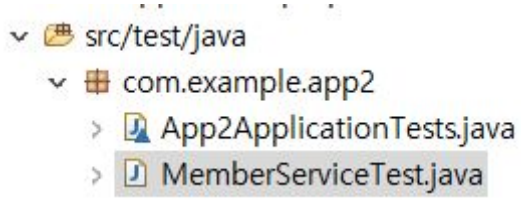




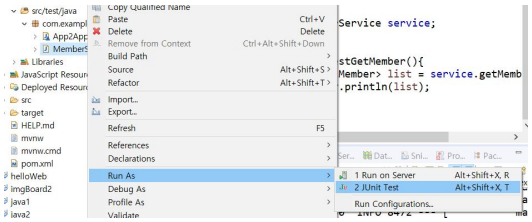




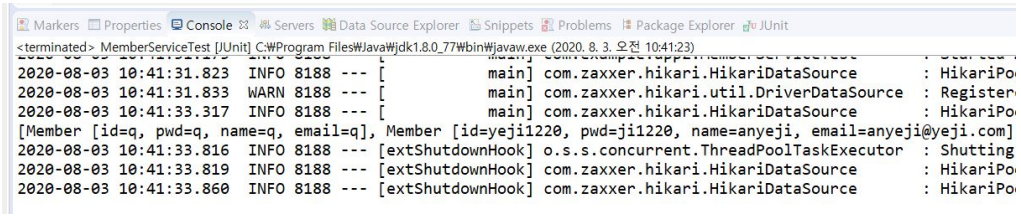
- src/test/java 패키지에 테스트 파일 생성

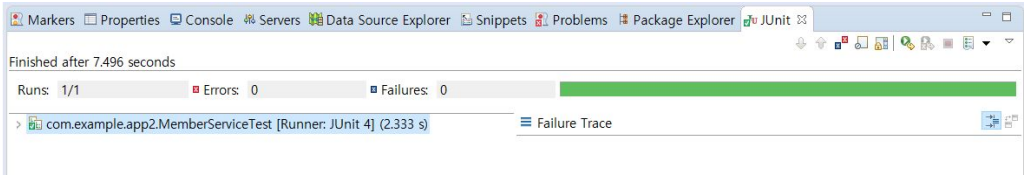


- RUN(J unit test로 실행)

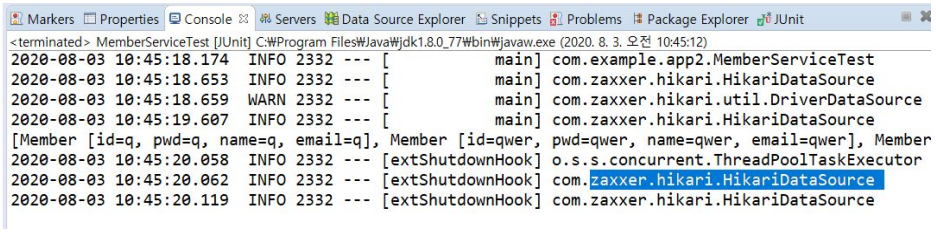


- 결과





*cf. data source: springboot가 hikari로 기본 설정



2. JPA(Java Persistence API)(Hibernate 기반) (db연동을 mybatis나 hibernate 둘 중 골라서 사용)
    - 새프로젝트 생성(app3)
    - pom.xml 생성 (mybatis 사용x)

app3/pom.xml | app2/pom.xml

```
27
28          <dependency>
29          <groupId>org.apache.tomcat.embed</groupId>
30          <artifactId>tomcat-embed-jasper</artifactId>
31          </dependency>
32          <dependency>
33          <groupId>javax.servlet</groupId>
34          <artifactId>jstl</artifactId>
35          </dependency>
36          <dependency>
37              <groupId>org.springframework.boot</groupId>
38              <artifactId>spring-boot-devtools</artifactId>
39              <optional>true</optional>
40          </dependency>
41          <dependency>
42              <groupId>com.oracle.database.jdbc</groupId>
43              <artifactId>ojdbc8</artifactId>
44              <scope>runtime</scope>
45          </dependency>
46
```

+JPA

app3/pom.xml | app2/pom.xml | application.properties | *User.java

```
26          </dependency>
27
28      <dependency>
29       <groupId>org.springframework.boot</groupId>
30       <artifactId>spring-boot-starter-data-jpa</artifactId>
31         </dependency>
```

+Thymeleaf

```
            <scope>runtime</scope>
        </dependency>
        <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
```

- application.properties 생성

app3/pom.xml | app2/pom.xml | application.properties

```
1 server.port=8888
2 #spring.main.web-application-type=none
3 #spring.mvc.view.prefix=/WEB-INF/views/
4 spring.mvc.view.suffix=.html
5
6 spring.datasource.driver-class-name=oracle.jdbc.driver.OracleDriver
7 spring.datasource.url=jdbc:oracle:thin:@localhost:1521/xe
8 spring.datasource.username=hr
9 spring.datasource.password=hr
10
11
12 spring.jpa.hibernate.ddl-auto=create
13 spring.jpa.database=oracle
14 spring.jpa.show-sql=true
```

+thymeleaf

```
spring.thymeleaf.prefix=classpath:templates/
spring.thymeleaf.check-template-location=true
spring.thymeleaf.suffix=.html
spring.thymeleaf.mode=HTML5
spring.thymeleaf.cache=false
spring.thymeleaf.order=0
```

*hibernate.ddl-auto: 수정할 때마다 reload *주의:데이터도 초기화 됨. 안쓸려면 주석처리

*show-sql : 작업 실행될 때마다 어케 작성되었는지 뿌려줌

- com.example.app3.model.join 패키지
  - User 클래스(dto)

    기존의 방식: vo와 동일한 이름의 컬럼으로 db에 table을 생성했었음.

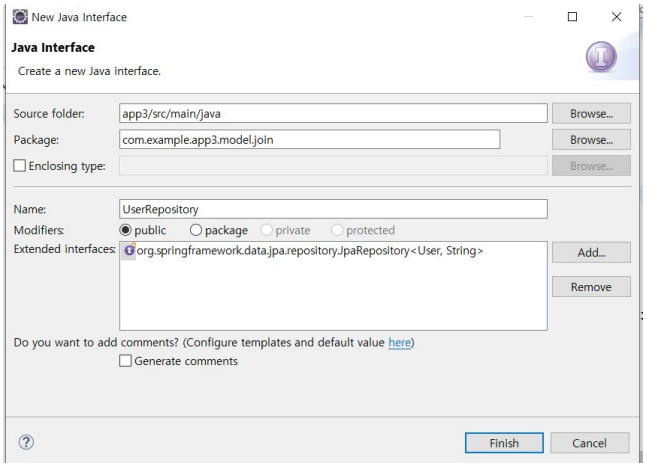
    현재의 방식: entity 어노테이션을 생성하면, db에 해당 멤버변수 속성을 가진 컬럼을 가진 table을 생성해준다. (기존의 member 테이블과 동일)

```
@Entity
public class Usertb {
    @Id
    private String id;
    private String pwd;
    private String name;
    private String email;

    //fetch: 속도 느려서 에러나는것 방지
    //mappedBy: 참조되는 컬럼
    //cascade: 참조된 row 삭제시 처리
    @OneToMany(fetch=FetchType.LAZY, mappedBy ="writer",cascade=CascadeType.REMOVE)
    private List<Article> articles;

    public Usertb() {
    }
    public Usertb(String id, String pwd, String name, String email) {
```

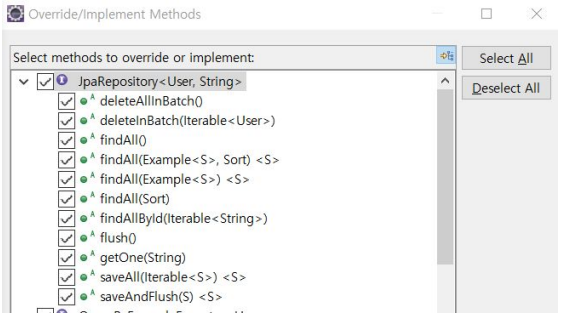
    *if 멤버변수와 테이블의 컬럼명 다르게 할 경우

```
@Entity
public class User {
    @Id
    @Column(name="user_id")
    private String id;
    private String pwd;
    private String name;
    private String email;
```

  - [dao/daoimpl] UserRepository.java(인터페이스) (JpaRepository 상속) <vo,primary key type>



해당 인터페이스로 daoimpl 역할 충족.
왜? 이미 부모클래스에서 daoimpl기능 구현되어있음. 끌어다 사용하면 됨



- [service] UserService.java(클래스)

```java
package com.example.app3.model.join;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class UserService {
    @Autowired
    private UserRepository repos;

    public void addUser(User u){
        repos.save(u);
    }
    public User getUser(String id){
        return repos.getOne(id);
    }
    public List<User> getUsers(){
        return repos.findAll();
    }
    public void editUser(User u){
        repos.save(u);
    }
    public void delUser(String id){
        repos.deleteById(id);
    }

}
```

- com.example.app3 패키지
  - [컨트롤러] UserController.java

```java
@Controller
public class UserController {
    @Autowired
    private UserService service;

    @GetMapping("/user/join")
    public String joinForm(){
        return "/user/join";
    }

    @PostMapping("/user/join")
    public String join(Usertb u){
        service.addUser(u);
        return "redirect:/user";
    }

    @GetMapping("/user")
    public String userList(Model m){
        List<Usertb> list = service.getUsers();
        System.out.println("list:"+list);
        m.addAttribute("list",list);
        return "/user/list";
    }

    @GetMapping("/user/{id}")//검색시 검색할 값을 함께 보냄. ex./user/asdf
    public String editForm(@PathVariable("id") String id, Model m){
        Usertb u = service.getUser(id);
        m.addAttribute("u", u);
        return "user/edit";
    }
```

```java
    @PostMapping("/user/edit")
    public String edit(Usertb u){
        service.editUser(u);
        return "redirect:/user";
    }

    @PostMapping("/user/delete")
    public String edit(@RequestParam("id") String id){
        service.delUser(id);
        return "redirect:/user";
    }

//  @GetMapping("/user/delete/{id}")
//  public String delForm(@PathVariable("id") String id){
//      service.delUser(id);
//      return "redirect:/user";
//  }
}
```

- view
  src/main/resources의 templates 디렉토리
  - /user/join.html

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="EUC-KR">
<title>Insert title here</title>
</head>
<body>
<h3>join form</h3>
<form action="/user/join" method="post">
id:<input type="text" name="id"><br>
pwd:<input type="text" name="pwd"><br>
name:<input type="text" name="name"><br>
email:<input type="text" name="email"><br>
<input type="submit" value="join"><br>
</form>
</body>
</html>
```

  - /user/list.html *thymeleaf 속성 사용(책 p.102~108. thymeleaf pom.xml에 추가)

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="EUC-KR">
<title>Insert title here</title>
</head>
<body>
<h3>user list</h3>
<table border="1">
<tr><th>id</th><th>pwd</th><th>name</th><th>email</th></tr>
<tr th:each="u : ${list}">
<td th:text="${u.id}"></td>
<td th:text="${u.pwd}"></td>
<td th:text="${u.name}"></td>
<td th:text="${u.email}"></td>
</table>
</body>
</html>
```

- /user/edit.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="EUC-KR">
<title>Insert title here</title>
<script type="text/javascript">
function del(){
    f.action="/user/delete"
    f.submit()
}
</script>
</head>
<body>
<h3>edit form</h3>
<form action="/user/edit" method="post" name="f">
id:<input type="text" name="id" th:value="${u.id}" readonly><br>
pwd:<input type="text" name="pwd" th:value="${u.pwd}"><br>
name:<input type="text" name="name" th:value="${u.name}" readonly><br>
email:<input type="text" name="email" th:value="${u.email}"><br>
<input type="submit" value="edit">
<input type="button" value="del" onclick="del()"><br>
</form>
</body>
</html>
```

- model.article 패키지
  - [dto] Article.java

```
@Entity
public class Article {
    @Id
    //자동 넘버링 타입: oracle은 시퀀스로 설정
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator="article_sequence")
    //시퀀스 생성
    @SequenceGenerator(name="article_sequence",sequenceName="seq_article")
    private int num;//글 번호 자동으로 넣어줄 경우 시퀀스 필요

    @ManyToOne//Usertb클래스와 다:1의 관계를 갖도록함
    @JoinColumn(name="writer",nullable=false)//name: writer를 조인, nullable: usertb에 없는 아이디는 글 작성 불가능하게
    private Usertb writer;//foreign key로 만들어주기 위해[article이 자식, usertb이 부모가 되도록 설정]

    private String title;
    private Date w_date;
    private String content;

    @PrePersist
    //쿼리 실행 전 실행: 쿼리문을 직접 작성해서 sysdate 쓰는 것이 아니기 때문에.
    //날짜를 생성해서 보낸다.
    public void beforeCreate(){
        w_date = new Date();
    }

    public Article() {
```

  - [daoimpl] ArticleRepository.java(인터페이스)

```java
package com.example.app3.model.article;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ArticleRepository extends JpaRepository<Article, Integer> {

}
```

- [Service] ArticleService.java [숙제]