

1. Spring Boot

1-1. 자바 11 설치

<https://blog.naver.com/qodlgks123/222616577711>

1-2. 스프링부트 프로젝트 시작

Gradle이나 Maven과 같은 빌드 도구를 이용할 수 있지만 조금 더 손쉽고 편한 방법으로 프로젝트를 만들기 위해 스프링 부트 프로젝트를 [Spring initializr](https://start.spring.io)를 이용
<https://start.spring.io>

The image shows the Spring Initializr web form. It is divided into two main sections: Project Metadata and Dependencies.

Project Metadata:

- Project:** ☐ Maven Project, ☒ Gradle Project
- Language:** ☒ Java, ☐ Kotlin, ☐ Groovy
- Spring Boot:** ☐ 3.0.0 (SNAPSHOT), ☐ 3.0.0 (M2), ☐ 2.7.0 (SNAPSHOT), ☐ 2.7.0 (M3), ☐ 2.6.7 (SNAPSHOT), ☒ 2.6.6, ☐ 2.5.13 (SNAPSHOT), ☐ 2.5.12
- Project Metadata:**
 - Group: com.thingspire
 - Artifact: vup_sample
 - Name: jyh
 - Description: Demo project for VUP
 - Package name: com.thingspire.vup_sample
 - Packaging: ☒ Jar, ☐ War
 - Java: ☐ 18, ☐ 17, ☒ 11, ☐ 8

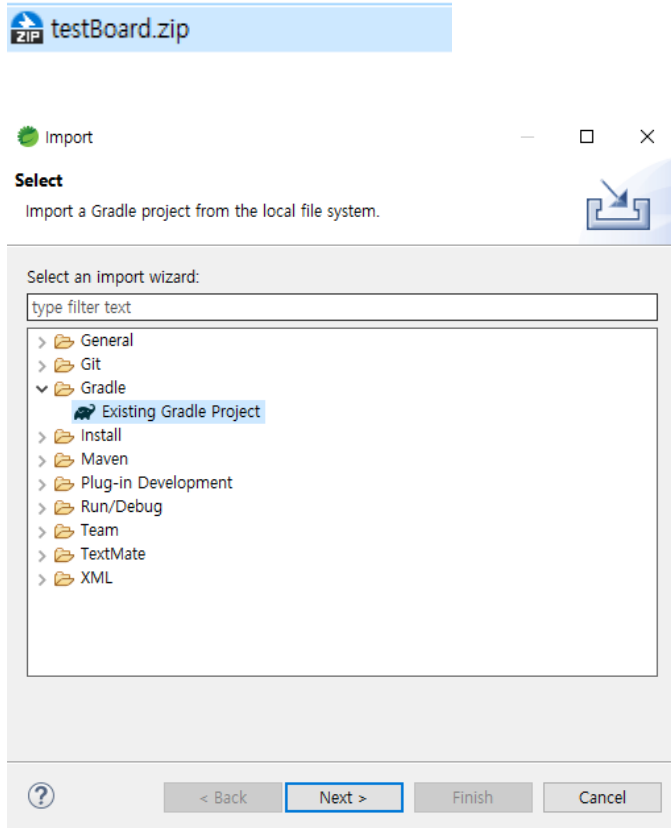
Dependencies:

- Spring Boot DevTools** (DEVELOPER TOOLS): Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
- Lombok** (DEVELOPER TOOLS): Java annotation library which helps to reduce boilerplate code.
- Spring Web** (WEB): Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
- Spring Data JPA** (SQL): Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
- MariaDB Driver** (SQL): MariaDB JDBC and R2DBC driver.

Buttons: ADD DEPENDENCIES... CTRL + B

- Group: 보통 기업의 도메인 명
- Artifact: 빌드 결과물 이름, 보통 프로젝트명
- Name: 프로젝트 이름
- Description: 프로젝트 설명
- Package name: 패키지 이름, 초기 소스들이 만들어질 기본 패키지명
- Packaging: 배포 형태 (war, jar)
- Java: 자바 버전 선택
- Dependencies : 이후에 설정 파일(ex: build.gradle)에 직접 등록해도 되지만, 프로젝트 생성 전에도 라이브러리를 미리 주입할 수 있다.

1-3. 프로젝트 임포트 및 실행

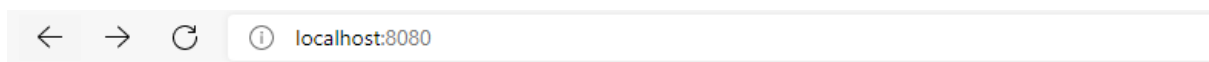


```
package com.example.testBoard;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class TestBoardApplication {

    public static void main(String[] args) {
        SpringApplication.run(TestBoardApplication.class, args);
    }
}
```



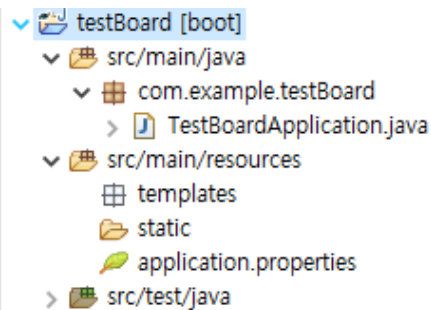
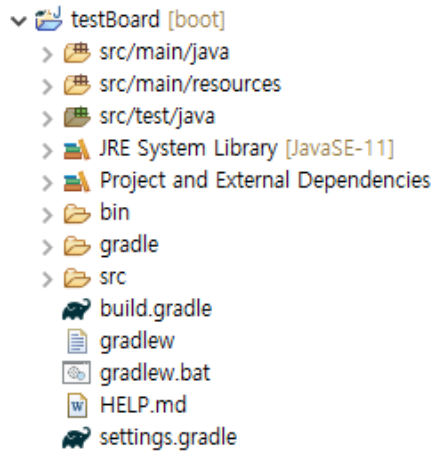
Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

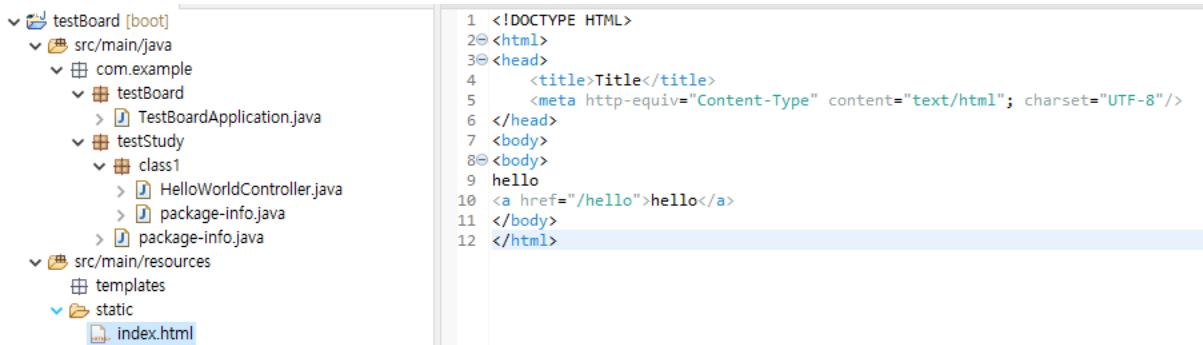
Thu Feb 10 14:55:12 KST 2022

There was an unexpected error (type=Not Found, status=404).

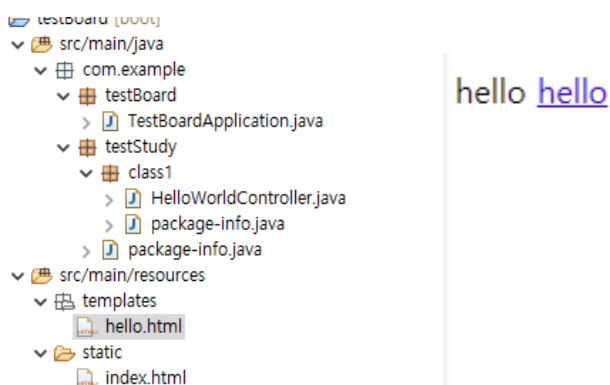
1-4. 테스트 프로젝트 구조



Static/index.html 웰컴페이지 기능



Controller.class 생성후 hello 페이지 이동



1-5. 타임리프

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org"> <!--타임리프 문법 사용하겠다는 표시-->
<head>
  <title>Title</title>
  <meta http-equiv="Content-Type" content="text/html"; charset="UTF-8"/>
</head>
<body>
<p th:text="'안녕하세요.' + ${data}">안녕하세요. 손님</p> <!--3-1의 컨트롤러에서 넘겨준 data를 받아서 화면에 처리-->
</body>
</html>
```

1-6. Controller.java 데이터 전달방식

```
public class HelloController {

    @GetMapping("hello") ///http://localhost:8080/hello
    public String hello(Model model) {
        model.addAttribute("data", "hello class2!!!");
        return "hello";
    }

    @GetMapping("hello-mvc") //http://localhost:8080/hello-mvc?name=spring!!!
    public String helloMvc(@RequestParam("name") String name, Model model) {
        model.addAttribute("name", name);
        return "hello-template";
    }

    @GetMapping("hello-string") //http://localhost:8080/hello-string?name=kkk
    @ResponseBody
    public String helloString(@RequestParam("name") String name) {
        return "hello " + name;
    }

    @GetMapping("hello-api")//http://localhost:8080/hello-api?name=kkk
    @ResponseBody
    public Hello helloApi(@RequestParam("name") String name) {
        Hello hello = new Hello();
        hello.setName(name);
        return hello;
    }
}
```

참고 : spring-boot-devtools 라이브러리 추가

1-7. 빌드

cmd 해당폴더 gradlew build

build - libs 에 가면 jar파일 생성됨 - 서버에 올리고 java -jar 해서 실행

4. 백엔드 스프링

4.1 class1

4.2 class2

```
package-info.java × HelloWorldController.java App.js build.gradle ×
1 plugins {
2     id 'org.springframework.boot' version '2.6.3'
3     id 'io.spring.dependency-management' version '1.0.11.RELEASE'
4     id 'java'
5 }
6
7 group = 'com.example'
8 version = '0.0.1-SNAPSHOT'
9 sourceCompatibility = '11'
10
11 repositories {
12     mavenCentral()
13 }
14
15 dependencies {
16     implementation 'org.springframework.boot:spring-boot-starter-web'
17     testImplementation 'org.springframework.boot:spring-boot-starter-test'
18 }
19
```

helloworldController

```
package-info.java × HelloWorldController.java × index.html build.gradle hello.html
1 package com.example.testBoard.testStudy.class2;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.ui.Model;
5 import org.springframework.web.bind.annotation.GetMapping;
6
7 @Controller // 메인시작
8 public class HelloWorldController {
9     @GetMapping("hello") // url hello
10    public String hello(Model model) {
11        model.addAttribute("data", "hello!!!MVC class2");
12        return "hello";
13    }
14 }
15
```

Templates/hello.html

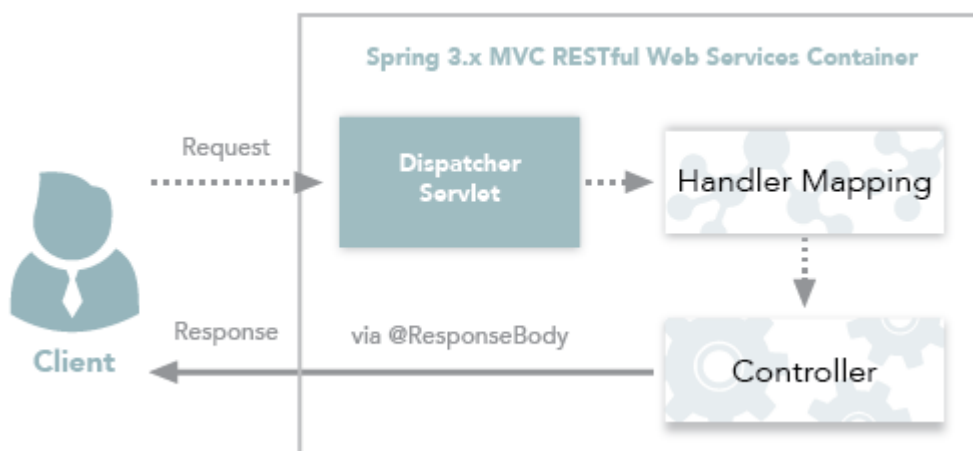
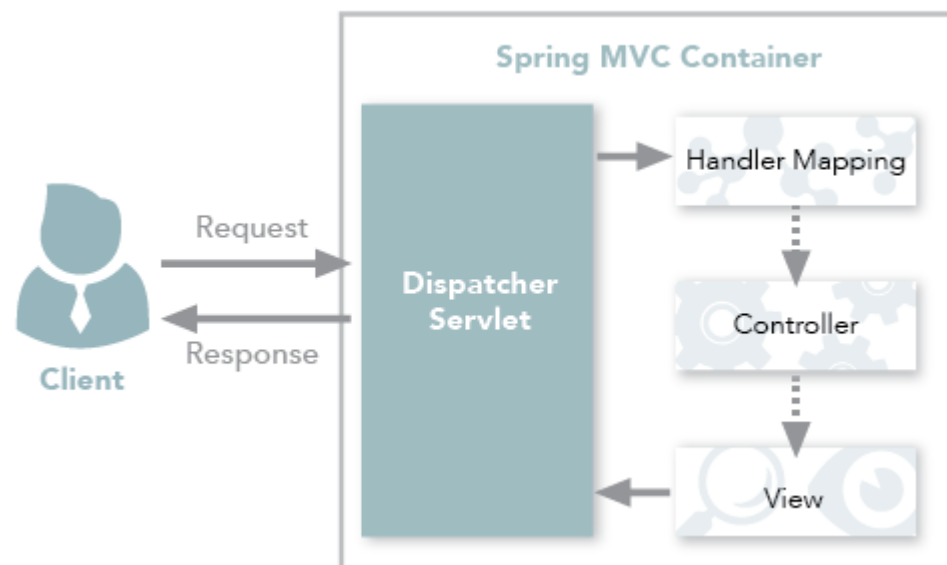
```
package-info.java HelloWorldController.java index.html build.gradle hello.html ×
1 <!DOCTYPE HTML>
2 <html xmlns:th="http://www.thymeleaf.org"> <!--타입리프 문법 사용하겠다는 표시-->
3 <head>
4     <title>Title</title>
5     <meta http-equiv="Content-Type" content="text/html"; charset="UTF-8"/>
6 </head>
7 <body>
8     <p th:text="'안녕하세요. ' + ${data}">안녕하세요. 손님</p> <!--3-1의 컨트롤러에서 넘겨준 data를 받아서 화면에 처리-->
9 </body>
10 </html>
```

4.3 RestController 과 Controller 차이

@RestController 은 @Controller 와 @ResponseBody 의 조합입니다.

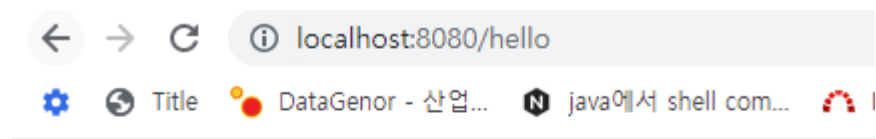
Spring 프레임 워크에서 RESTful 웹 서비스를 보다 쉽게 개발할 수 있도록 Spring 4.0 에서 추가되었습니다.

근본적인 차이점은 @Controller 의 역할은 Model 객체를 만들어 데이터를 담고 View 를 찾는 것이지만, @RestController 는 단순히 객체만을 반환하고 객체 데이터는 JSON 또는 XML 형식으로 HTTP 응답에 담아서 전송합니다. 물론 @Controller 와 @ResponseBody 를 사용하여 만들 수 있지만 이러한 방식은 RESTful 웹서비스의 기본 동작이기 때문에 Spring 은 @Controller 와 @ResponseBody 의 동작을 조합한 @RestController 을 도입했습니다.

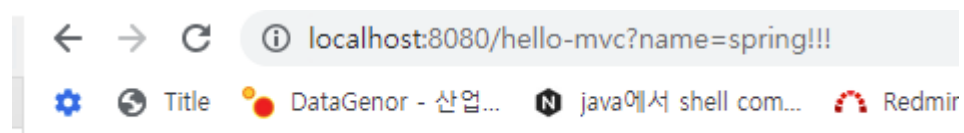


4-4. class2 – 소스참조

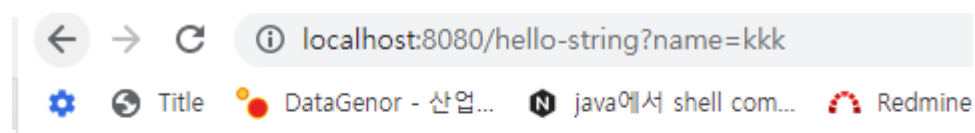
```
runtimeOnly('org.springframework.boot:spring-boot-devtools')
```



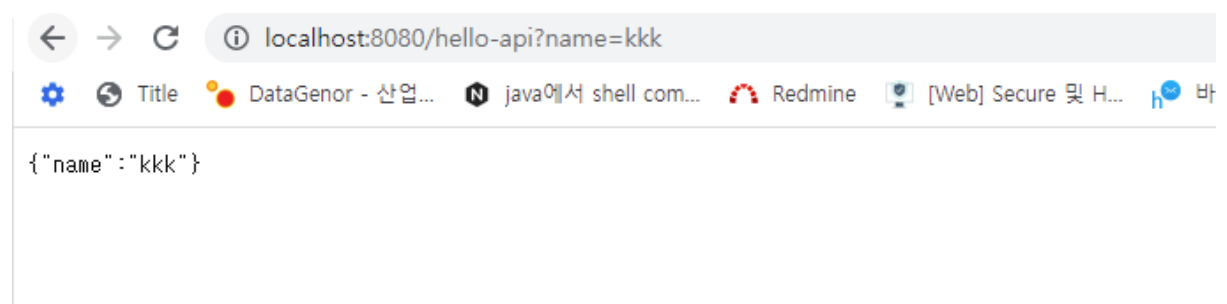
안녕하세요.hello class2!!!



안녕하세요. spring!!!

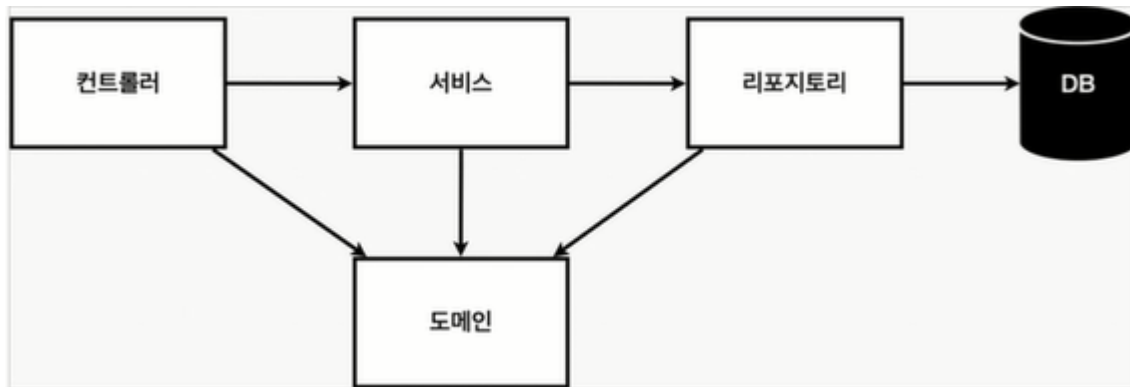


hello kkk



```
{"name":"kkk"}
```

4-5. 회원관리 예제 - 백앤드



```
package-info.java Member.java ×
1 package com.example.testBoard.testStudy.class3.domain;
2
3 public class Member {
4
5     private Long id;
6     private String name;
7
8     public Long getId() {
9         return id;
10    }
11    public void setId(Long id) {
12        this.id = id;
13    }
14    public String getName() {
15        return name;
16    }
17    public void setName(String name) {
18        this.name = name;
19    }
20 }
```

```
package-info.java × Member.java MemberRepository.java ×
1 package com.example.testBoard.testStudy.class3.repository;
2
3 import java.util.List;
4
5 public interface MemberRepository {
6     Member save(Member member);
7     Optional<Member> findById(Long id);
8     Optional<Member> findByName(String name);
9     List<Member> findAll();
10 }
11 }
```



```

package-info.java × Member.java MemberRepository.java MemoryMemberRepository.java ×
1 package com.example.testBoard.testStudy.class3.repository;
2
3 import java.util.ArrayList;
4
5 public class MemoryMemberRepository implements MemberRepository {
6
7     private static Map<Long, Member> store = new HashMap<>();
8     private static Long sequence = 0L;
9
10    @Override
11    public Member save(Member member) {
12        member.setId(++sequence);
13        store.put(member.getId(), member);
14        return member;
15    }
16
17    @Override
18    public Optional<Member> findById(Long id) {
19        return Optional.ofNullable(store.get(id));
20    }
21
22    @Override
23    public Optional<Member> findByName(String name) {
24        return store.values().stream().filter(member -> member.getName().equals(name)).findAny();
25    }
26
27    @Override
28    public List<Member> findAll() {
29        return new ArrayList<>(store.values());
30    }
31
32    public void clearStore() {
33        store.clear();
34    }
35 }

```

```

package-info.java × Member.java MemberRepository.java MemoryMemberRepository.java MemberService.java ×
1 package com.example.testBoard.testStudy.class3.service;
2
3 import java.util.List;
4
5 public class MemberService {
6
7     //private final MemberRepository memberRepository = new MemoryMemberRepository();
8     private final MemberRepository memberRepository;
9
10    public MemberService(MemoryMemberRepository memberRepository) {
11        this.memberRepository = memberRepository;
12    }
13
14    public Long join(Member member) {
15        // 같은 이름이 있는 중복회원X
16        // 1
17        /*
18         * Optional<Member> result = memberRepository.findByName(member.getName());
19         * result.ifPresent(m -> { throw new IllegalStateException("이미 존재하는 회원입니다.");
20         * });
21         */
22
23        // 2
24        validateDuplicateMember(member);
25        memberRepository.save(member);
26        return member.getId();
27    }
28
29    //2. optional을 get으로 바로 반환하는것보다 아래처럼 쓰는게 좋다.
30    private void validateDuplicateMember(Member member) {
31        memberRepository.findByName(member.getName()).ifPresent(m -> {
32            throw new IllegalStateException("이미 존재하는 회원입니다.");
33        });
34    }
35
36    public List<Member> findMembers() {
37        return memberRepository.findAll();
38    }
39
40    public Optional<Member> findOne(Long memberId) {
41        return memberRepository.findById(memberId);
42    }
43 }

```

4-6. junit

```
package-info.java Member.java MemberRepository.java MemoryMemberRepository.java MemberService.java MemoryMemberRepositoryTest.java x
13 public class MemoryMemberRepositoryTest {
14
15     MemoryMemberRepository repository = new MemoryMemberRepository();
16
17     @AfterEach
18     public void afterEach() {
19         repository.clearStore();
20     }
21
22     @Test
23     public void save() {
24         Member member = new Member();
25         member.setName("Spring");
26         repository.save(member);
27         Member result = repository.findById(member.getId()).get();
28         // System.out.println("" + (result == member));
29         // Assertions.assertThat(member).isEqualTo(result);
30         // Assertions.assertEquals(member, result);
31         assertThat(member).isEqualTo(result);
32     }
33
34     @Test
35     public void findByName() {
36         Member member1 = new Member();
37         member1.setName("spring1");
38         repository.save(member1);
39
40         Member member2 = new Member();
41         member2.setName("spring2");
42         repository.save(member2);
43
44         Member result = repository.findByName("spring1").get();
45         assertThat(result).isEqualTo(member1);
46     }
47
48     @Test
49     public void findAll() {
50         Member member1 = new Member();
51         member1.setName("spring1");
52         repository.save(member1);
53
54         Member member2 = new Member();
55         member2.setName("spring2");
56         repository.save(member2);
57
58         List<Member> result = repository.findAll();
59         assertThat(result.size()).isEqualTo(2); //이전 test에 repo에 저장해서 안맞게 결과가 나온 -> clearStore 꼭 해줘야 동작
60     }
61 }
```

```
package-info.java Member.java MemberRepository.java MemoryMemberRepository.java MemberService.java MemoryMemberRepositoryTest.java x
16 MemberService memberService;
17 MemoryMemberRepository memberRepository;
18
19 @BeforeEach
20 public void beforeEach() {
21     memberRepository = new MemoryMemberRepository();
22     memberService = new MemberService(memberRepository); //di
23 }
24
25 /*
26  * @AfterEach public void afterEach() { memberRepository.clearStore(); }
27  */
28
29 @Test
30 void 회원가입() {
31     // given
32     Member member = new Member();
33     member.setName("hello");
34
35     // when
36     Long svaeId = memberService.join(member);
37
38     // then
39     Member findeMember = memberService.findOne(svaeId).get();
40     // Optional<findeMember> = memberService.findOne(svaeId);
41     assertThat(member.getName()).isEqualTo(findeMember.getName());
42 }
43
44 @Test
45 public void 중복_회원_예외() {
46     // given
47     Member member1 = new Member();
48     member1.setName("spring1");
49
50     Member member2 = new Member();
51     member2.setName("spring1");
52
53     // when
54     memberService.join(member1);
55     IllegalStateException e = assertThrows(IllegalStateException.class, () -> memberService.join(member2));
56     assertThat(e.getMessage()).isEqualTo("이미 존재하는 회원입니다.");
57     // assertThrows(IllegalStateException.class, () -> memberService.join(member2));
58     /*
59     * memberService.join(member1); try { memberService.join(member2); fail(); }
60     * catch (IllegalStateException e) {
61     *     assertThat(e.getMessage()).isEqualTo("이미 존재하는 회원입니다."); }
62     */
63 }
```

4-7. 스프링빈 의존성설정

```
Member.java MemberService.java MemberRepository.java MemoryMemberRepository.java package-info.java MemberController.java X
1 package com.example.testBoard.testStudy.class4.controller;
2
3 import java.util.List;
4
5 @Controller
6 public class MemberController {
7     private final MemberService memberService;
8     // @Autowired private MemberService memberService; //DI - 필드주입(중간에 바꿀 방법이 없음)
9
10    /*
11     * @Autowired public void setMemberService(MemberService memberService) {
12     *     this.memberService = memberService; } //DI - setter주입 (누군가가 memberController를
13     *     호출할때 public으로 되어있어야함. 따라서 변경위험성 있음)
14     */
15    @Autowired
16    public MemberController(MemberService memberService) {
17        this.memberService = memberService;
18    } // DI- 생성자주입 (결론적으로 이걸 쓰는게 좋음. 의존관계가 실행중에 동적으로 변하는 경우는 거의없다)
19
20    @GetMapping("/members/new")
21    public String createForm() {
22        return "members/createMemberForm"; // templates에 html호출
23    }
24
25    @PostMapping("/members/new")
26    public String create(MemberForm form) { // 화면에서 버튼을 누르면 name값을 받아와 메서드 실행
27        Member member = new Member();
28        member.setName(form.getName());
29
30        memberService.join(member);
31
32        return "redirect:/"; // 원래화면으로 돌아감
33    }
34
35    @GetMapping("/members")
36    public String list(Model model) {
37        List<Member> members = memberService.findMembers();
38        model.addAttribute("members", members);
39        return "members/memberList";
40    }
41 }
```

4-8. 환경설정

```
MemberRepository.java package-info.java SpringConfig.java X
1 package com.example.testBoard.testStudy.class4;
2
3 import org.springframework.context.annotation.Bean;
4
5 @Configuration
6 public class SpringConfig {
7
8     @Bean
9     public MemberService memberService() {
10         return new MemberService(memberRepository());
11     }
12
13     @Bean
14     public MemberRepository memberRepository() {
15         return new MemoryMemberRepository();
16     }
17 }
18
19
20
21
22
23
```

4-9. DI

```
@Controller
public class MemberController {
    private final MemberService memberService;
    //@Autowired private MemberService memberService; //DI - 필드주입 (중간에 바꿀 방법이 없음)

    /*
    @Autowired
    public void setMemberService(MemberService memberService) {
        this.memberService = memberService;
    }*/ //DI - setter주입 (누군가가 memberController를 호출할때 public으로 되어있어야함. 따라서 변경위험성 있음)

    @Autowired
    public MemberController(MemberService memberService) {
        this.memberService = memberService;
    } //DI- 생성자주입 (결론적으로 이걸 쓰는게 좋음. 의존관계가 실행중에 동적으로 변하는 경우는 거의없다)
}
```

MemberRepository.java × package-info.java SpringConfig.java MemberController.java HomeController.java ×

```
1 package com.example.testBoard.testStudy.class4.controller;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.GetMapping;
5
6 @Controller
7 public class HomeController {
8
9     @GetMapping("/")
10    public String home() {
11        return "home";
12    }
13 }
14
```

MemberRepository.java × package-info.java SpringConfig.java home.html ×

```
1 <!DOCTYPE HTML>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <body>
4
5 <div class="container">
6     <div>
7         <h1>hello spring</h1>
8         <p>회원기능</p>
9         <p>
10             <a href="/members/new">회원가입</a>
11             <a href="/members">회원목록</a>
12         </p>
13     </div>
14 </div>
15 </body>
16 </html>
```

hello spring

회원기능

[회원가입](#) [회원목록](#)

4-10. 메모리 기본 cr

```
MemberRepository.java package-info.java createMemberForm.html memberList.html
1 <!DOCTYPE HTML>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <body>
4
5 <div class="container">
6   <form action="/members/new" method="post">
7     <div class="form-group">
8       <label for="name">이름</label>
9       <input type="text" id="name" name="name" placeholder="이름을 입력하세요">
10    </div>
11    <button type="submit">등록</button>
12  </form>
13 </div>
14 </body>
15 </html>
```

```
MemberRepository.java package-info.java createMemberForm.html memberList.html
1 <!DOCTYPE HTML>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <body>
4
5 <div class="container">
6   <div>
7     <table>
8       <thead>
9         <tr>
10          <th>#</th>
11          <th>이름</th>
12        </tr>
13      </thead>
14      <tbody>
15        <tr th:each="member : ${members}">
16          <td th:text="${member.id}"></td>
17          <td th:text="${member.name}"></td>
18        </tr>
19      </tbody>
20    </table>
21  </div>
22 </div>
23 </body>
24 </html>
```

← → ↻ ① localhost:8080/members/new

이름

← → ↻ ① localhost:8080/members

이름

1 123123


2 123123435

3 dfgdgvf

4 324324

4-11. 디비 연동 – h2 db

<https://www.h2database.com>



Translate
Search:

Home
Download
Cheat Sheet

Documentation
Quickstart
Installation
Tutorial
Features
Security
Performance
Advanced

H2 Database Engine

Welcome to H2, the Java SQL database. The main features of H2 are:

- Very fast, open source, JDBC API
- Embedded and server modes: in-memory databases
- Browser based Console application
- Small footprint: around 2.5 MB jar file size

Download

Version 2.1.210 (2022-01-17)

- Windows Installer (6.7 MB)
- All Platforms (zip, 9.5 MB)

[All Downloads](#)

Support

[Stack Overflow \(tag H2\)](#)

[Google Group](#)

For non-technical issues, use: [dbsupport at h2database.com](mailto:dbsupport@h2database.com)

Features

```
C:\Program Files (x86)\H2\bin>h2.sh
```

로그인

저장한 설정: Generic H2 (Embedded)

설정 이름: Generic H2 (Embedded) 저장 삭제

드라이버 클래스: org.h2.Driver

JDBC URL: jdbc:h2:~/test

사용자명: sa

비밀번호:

연결 연결 시험

로그인

저장한 설정: Generic H2 (Server)

설정 이름: Generic H2 (Server) 저장 삭제

드라이버 클래스: org.h2.Driver

JDBC URL: jdbc:h2:tcp://localhost/~ /test

사용자명: sa

비밀번호:

연결 연결 시험

실행 | Run Selected | 자동 완성 | 지우기 | SQL 문:

```
create table member
(
  id bigint generated by default as identity,
  name varchar(255),
  primary key (id)
);
```

1-2. build.gradle

```
implementation 'org.springframework.boot:spring-boot-starter-jdbc'
runtimeOnly 'com.h2database:h2'
```

1-3. application.properties

```
spring.datasource.url=jdbc:h2:tcp://localhost/~ /test
spring.datasource.driver-class-name=org.h2.Driver
```

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.boot:spring-boot-starter-jdbc'

    runtimeOnly('org.springframework.boot:spring-boot-devtools')
    runtimeOnly 'com.h2database:h2'

    testImplementation('org.springframework.boot:spring-boot-starter-test') {
        exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
    }
}
```

← → ↻ ⓘ localhost:8080/members

⚙️ 🌐 Title 🟡 DataGenor - 산업... 🟡 java에서 shell com... 🔴 F

이름

1 abc

2 abc

3 abc

4 cdf

5 r342fewdf

select * from member;

select * from member;

ID	NAME
1	abc
2	abc
3	abc
4	cdf
5	r342fewdf

package-info.java × JpaMemberRepository.java build.gradle application.properties Member.java ×

```
1 package com.example.testBoard.testStudy.class5.domain;
2
3 import javax.persistence.Column;
4 import javax.persistence.Entity;
5 import javax.persistence.GeneratedValue;
6 import javax.persistence.GenerationType;
7 import javax.persistence.Id;
8
9 @Entity
10 public class Member {
11
12     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
13     private Long id;
14     // @Column(name="username")
15     private String name;
16
17     public Long getId() {
18         return id;
19     }
20     public void setId(Long id) {
21         this.id = id;
22     }
23     public String getName() {
24         return name;
25     }
26     public void setName(String name) {
27         this.name = name;
28     }
29 }
30
```

4-12. Repository교체

4-12-1. Jdbc

```
package-info.java  JdbcMemberRepository.java X
1 package com.example.testBoard.testStudy.class5.repository;
2
3 import java.sql.Connection;
4
5 //귀찮 대충 하나만
6 public class JdbcMemberRepository implements MemberRepository {
7
8     private final DataSource datasource;
9
10    public JdbcMemberRepository(DataSource datasource) {
11        this.datasource = datasource;
12    }
13
14    private Connection getConnection() {
15        return DataSourceUtils.getConnection(datasource); //이걸 써야 커넥션이 여러개 안생김
16    }
17
18    @Override
19    public Member save(Member member) {
20        String sql = "insert into member(name) values(?)";
21
22        Connection conn = null;
23        PreparedStatement pstmt = null;
24        ResultSet rs = null;
25
26        try {
27            conn = getConnection();
28            pstmt = conn.prepareStatement(sql, java.sql.Statement.RETURN_GENERATED_KEYS);
29
30            pstmt.setString(1, member.getName());
31            pstmt.executeUpdate();
32
33            rs = pstmt.getGeneratedKeys();
34
35            if (rs.next()) {
36                member.setId(rs.getLong(1));
37            } else {
38                throw new SQLException("id 조회실패");
39            }
40        } catch (Exception e) {
41            e.printStackTrace();
42        } finally {
43            conn.close();
44            pstmt.close();
45            rs.close();
46        }
47    }
48
49 }
```

4-12-2. JdbcTemplate

```
package-info.java X  JdbcMemberRepository.java  JdbcTemplateMemberRepository.java X
1 package com.example.testBoard.testStudy.class5.repository;
2
3 import java.util.HashMap;
4
5 public class JdbcTemplateMemberRepository implements MemberRepository {
6
7     private final JdbcTemplate jdbcTemplate;
8
9     @Autowired // 생성자 하나면 autowired생략가능
10    public JdbcTemplateMemberRepository(DataSource dataSource) {
11        jdbcTemplate = new JdbcTemplate(dataSource);
12    }
13
14    @Override
15    public Member save(Member member) {
16        SimpleJdbcInsert jdbcInsert = new SimpleJdbcInsert(jdbcTemplate);
17        jdbcInsert.withTableName("member").usingGeneratedKeyColumns("id");
18
19        Map<String, Object> parameters = new HashMap<>();
20        parameters.put("name", member.getName());
21
22        Number key = jdbcInsert.executeAndReturnKey(new MapSqlParameterSource(parameters));
23        member.setId(key.longValue());
24        return member;
25    }
26
27    @Override
28    public Optional<Member> findById(Long id) {
29        List<Member> result = jdbcTemplate.query("select * from member where id = ?", memberRowMapper(), id);
30        return result.stream().findAny();
31    }
32
33    @Override
34    public Optional<Member> findByName(String name) {
35        List<Member> result = jdbcTemplate.query("select * from member where name = ?", memberRowMapper(), name);
36        return result.stream().findAny();
37    }
38
39 }
```


4-12-3. Jpa

```
package-info.java JdbcMemberRepository.java JdbcTemplateMemberRepository.java JpaMemberRepository.java X
1 package com.example.testBoard.testStudy.class5.repository;
2
3 import java.util.List;
4
5 public class JpaMemberRepository implements MemberRepository {
6
7     private EntityManager em; // 이제 데이터소스 커넥션을 다 들고있음
8
9     public JpaMemberRepository(EntityManager em) {
10         this.em = em;
11     }
12
13     @Override
14     public Member save(Member member) {
15         em.persist(member);
16         return member;
17     }
18
19     @Override
20     public Optional<Member> findById(Long id) {
21         Member member = em.find(Member.class, id);
22         return Optional.ofNullable(member);
23     }
24
25     @Override
26     public Optional<Member> findByName(String name) {
27         List<Member> result = em.createQuery("select m from Member m where m.name = :name", Member.class)
28             .setParameter("name", name).getResultList();
29         return result.stream().findAny();
30     }
31
32     @Override
33     public List<Member> findAll() {
34         return em.createQuery("select m from Member m", Member.class).getResultList();
35     }
36 }
37
38 }
```

4-12-4. SpringDataJpa

```
package-info.java X JdbcMemberRepository.java JdbcTemplateMemberRepository.java JpaMemberRepository.java SpringDataJpaMemberRepository.java X
1 package com.example.testBoard.testStudy.class5.repository;
2
3 import java.util.Optional;
4
5 public interface SpringDataJpaMemberRepository extends JpaRepository<Member, Long>, MemberRepository {
6
7     @Override
8     Optional<Member> findByName(String name);
9 }
10
11 }
```

4-13. Aop

```

package-info... JdbcMemberRe... JdbcTemplat... JpaMemberRep... SpringDataJp... TimeTraceAop... X HomeControll... SpringConfig... TestBoardApp...
1 package com.example.testBoard.testStudy.class5.aop;
2
3 import org.aspectj.lang.ProceedingJoinPoint;
4 import org.aspectj.lang.annotation.Around;
5 import org.aspectj.lang.annotation.Aspect;
6 import org.springframework.stereotype.Component;
7
8 @Aspect
9 @Component
10 public class TimeTraceAop {
11
12     @Around("execution(* com.example.testBoard.testStudy.class5.controller.HomeController.home())")
13     public Object execute(ProceedingJoinPoint joinPoint) throws Throwable {
14         long start = System.currentTimeMillis();
15         System.out.println("start : " + joinPoint.toString());
16         try {
17             return joinPoint.proceed();
18         } finally {
19             long finish = System.currentTimeMillis();
20             long timeMs = finish - start;
21             System.out.println("end : " + joinPoint.toString());
22             System.out.println("ms : " + timeMs);
23         }
24     }
25 }
26

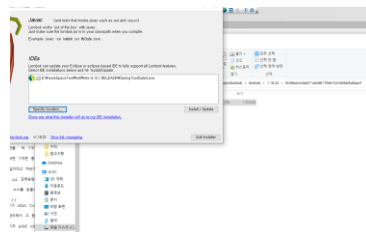
```

```

start : execution(String com.example.testBoard.testStudy.class5.controller.HomeController.home())
end : execution(String com.example.testBoard.testStudy.class5.controller.HomeController.home())
ms : 5
start : execution(String com.example.testBoard.testStudy.class5.controller.MemberController.list(Model))
start : execution(List com.example.testBoard.testStudy.class5.service.MemberService.findMembers())
start : execution(List org.springframework.data.jpa.repository.JpaRepository.findAll())
Hibernate: select member0_.id as id1_0_, member0_.name as name2_0_ from member member0_
end : execution(List org.springframework.data.jpa.repository.JpaRepository.findAll())
ms : 265
end : execution(List com.example.testBoard.testStudy.class5.service.MemberService.findMembers())
ms : 273
end : execution(String com.example.testBoard.testStudy.class5.controller.MemberController.list(Model))
ms : 311

```

4-14. Lombok



```

package-inf... JdbcMemberRe... JdbcTemplat... JpaMemberRe... SpringConfig... TestBoardApp... Member.java X
1 package com.example.testBoard.testStudy.class5.domain;
2
3 import javax.persistence.Column;
4
5 @Entity
6 @Data
7 public class Member {
8
9     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
10     private Long id;
11     @Column(name="username")
12     private String name;
13
14     /*
15     public Long getId() {
16         return id;
17     }
18     public void setId(Long id) {
19         this.id = id;
20     }
21     public String getName() {
22         return name;
23     }
24     public void setName(String name) {
25         this.name = name;
26     }
27 */
28 }
29

```

스웨거 3.x적용

스프링폭스- 2.6은 지원안함

스프링독스 이거사용

<https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-ui>

implementation 'org.springdoc:springdoc-openapi-ui:1.6.6'

6. 사용자 관리 화면

6-1. 의존성 추가 및 mysql 설정

implementation'mysql:mysql-connector-java'

```
# Jpa 설정
# true 설정시 JPA 쿼리문 확인 가능
spring.jpa.show-sql=true
# DDL(create, alter, drop) 정의시 DB의 고유 기능을 사용할 수 있다.
spring.jpa.hibernate.ddl-auto=update
# JPA의 구현체인 Hibernate가 동작하면서 발생한 SQL의 가독성을 높여준다.
spring.jpa.properties.hibernate.format_sql=true

# MySQL 설정
# Driver
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
# DB Source URL
spring.datasource.url=jdbc:mysql://192.168.10.99:3306/ ?useSSL=false&useUnicode=true&serverTimezone=Asia/Seoul
# DB username
spring.datasource.username=
# DB password
spring.datasource.password=
```

6-2. 롬복 및 스프링데이터 jpa

```
2022-02-23 14:20:02.904 INFO 15512 --- [ restartedmain] org.hibernate.u
Hibernate:
```

```
create table user (
  usid varchar(255) not null,
  before_uspw varchar(255),
  dept varchar(255),
  duty_c varchar(255),
  email varchar(255),
  gl_cd varchar(255),
  ip varchar(255),
  locked varchar(255),
  name varchar(255),
  new_token varchar(255),
  otp varchar(255),
  otp_check varchar(255),
  otp_use_yn varchar(255),
  out_admin_code varchar(255),
  out_admin_usid varchar(255),
  phone_number varchar(255),
  pos_c varchar(255),
  prj_id varchar(255),
  proxy_end_dt varchar(255),
  proxy_start_dt varchar(255),
  .....
```

```
package-info.java × UserRepository.java ×
1 package com.example.testBoard.sample.userManagement.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5
6
7
8 @Repository
9 public interface UserRepository extends JpaRepository<User, Long>{
10 }
11
```

6-3. jpa 메서드 및 junit 테스트

- save : entity를 저장하는 메소드(insert, update)
- flush : EntityManager의 내용을 DB에 동기화하는 메소드
- saveAndFlush : entity에 대한 저장 작업 후 flush
- delete : entity를 삭제하는 메소드(delete)
- deleteAll : DB의 모든 레코드를 persistence context로 읽어와 삭제하는 메소드
- deleteInBatch : persistence context로 읽어오지 않고 DB의 모든 레코드를 삭제하는 메소드
- findOne : primary key로 DB에서 Entity를 찾아오는 메소드(select)
- findAll : 모든 entity를 찾아오는 메소드(select)
- exists : primary key에 해당하는 entity가 존재하는 확인하는 메소드
- count : entity의 갯수를 확인하는 메소드

The screenshot shows an IDE with a Java file named `UserServiceTest`. The code includes a `@Transactional` annotation and a `@Autowired` dependency on `userRepository`. It contains two test methods: `사용자추가()` (add user) and `사용자조회()` (find user). The `사용자추가()` method creates a `User` object with specific test data, saves it, and asserts that the saved user's name matches the original. The `사용자조회()` method creates an admin user, finds it by ID, prints its details, asserts the name, and then lists all users.

Below the code editor, the JUnit runner interface is visible. It shows that the test suite `UserServiceTest` has completed successfully. The status bar indicates `Runs: 4/4`, `Errors: 0`, and `Failures: 0`. A green progress bar is shown at the bottom of the runner window.

```
18 @Transactional
19 public class UserServiceTest {
20     @Autowired
21     UserRepository userRepository;
22
23     @Test
24     void 사용자추가() {
25         User user = new User();
26         user.setUsid("testUser01");
27         user.setName("테스트유저01");
28         user.setDept("1");
29         user.setTeam("1");
30         user.setLocked("N");
31         user.setPwfailcnt(0);
32
33         User saveUser = userRepository.save(user);
34         assertThat(user.getName()).isEqualTo(saveUser.getName());
35     }
36
37     @Test
38     void 사용자조회() {
39         User user = new User();
40         user.setUsid("admin");
41         user.setName("어드민");
42
43         Optional<User> findUser = userRepository.findById(user.getUsid());
44         //System.out.println(findUser.toString());
45         assertThat(user.getName()).isEqualTo(findUser.get().getName());
46
47         List<User> userList = userRepository.findAll();
48         userList.stream().forEach(s -> System.out.println(s));
49     }
}
```

JUnit 5

Finished after 6.649 seconds

Runs: 4/4 Errors: 0 Failures: 0

> UserServiceTest [Runner: JUnit 5] (0.778 s)

Failure Trace

6-4. 현재 Package.json

```
{
  "name": "webapp",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.2",
    "@testing-library/react": "^12.1.2",
    "@testing-library/user-event": "^13.5.0",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-scripts": "5.0.0",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
```

```

    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "proxy": "http://localhost:8080"
}

```

6-5. toast-ui적용

npm install --save @toast-ui/react-grid

종속성에러발생시

npm install --save --legacy-peer-deps @toast-ui/react-grid

출처 : [tui.grid/패키지/토스트 ui.react-grid at master · nhn/tui.grid · 기트허브 \(github.com\)](https://github.com/nhn/tui.grid)

App.js추가

```
import 'tui-grid/dist/tui-grid.css';
```

```
import Grid from '@toast-ui/react-grid';
```

```
const data = [  
  {id: 1, name: 'Editor'},  
  {id: 2, name: 'Grid'},  
  {id: 3, name: 'Chart'}  
];
```

```
const columns = [  
  {name: 'id', header: 'ID'},  
  {name: 'name', header: 'Name'}  
];
```

```
const MyComponent = () => (  
  <Grid  
    data={data}  
    columns={columns}  
    rowHeight={25}  
    bodyHeight={100}  
    heightResizable={true}  
    rowHeaders={['rowNum']}  
  />  
);
```

```
////////////////////////////////
```

```
class MyComponent extends React.Component {  
  gridRef = React.createRef();  
  
  handleAppendRow = () => {  
    this.gridRef.current.getInstance().appendRow({});  
  };  
  
  render() {  
    return (  
      <>  
        <Grid ref={this.gridRef} data={data} columns={columns} />  
        <button onClick={this.handleAppendRow}>Append Row</button>  
      </>  
    );  
  }  
}
```

MyComponent -> App, React.Component -> Component 변경

```
////class7////
```

```

@RestController
@RequestMapping("/user")
public class UserController {
    private final UserService userService;

    @Autowired
    public UserController(UserService userService) {
        this.userService = userService;
    }

    @GetMapping
    public List<User> getUserList() {
        System.out.println("유저 리스트 출력");
        return userService.readAll();
    }
}

```

localhost:8080/user

← → ↺ 🌐 local... DataGenor - 산업... N java에서 shell com... Redmine [Web] Secure 및 H... 🗨️ 바넷정보기술 📄 애드몹(AdMob) 열... 📱 APK 분석하기 위한... [상태크 따라하기]...

```

[{"uid":"041022","name":"박기영","uspw":"6b86b273f134fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dept":"24","team":"40","email":"dfghg123@bisl.co.kr","role_ty":
00:00:00","phone_number":"","rank_cd":"","pos_cd":"","duty_cd":"","pfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","u
00:00:00","proxy_end_dt":"0000-00-00 00:00:00","gl_cd":null},{uid":"10","name":"김사장","uspw":"6b86b273f134fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dept":"
00:00:00","uspw_chng_dt":"2099-07-26
00:00:00","phone_number":"","rank_cd":"A_21","pos_cd":"","duty_cd":"1002","pfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user_ty
00:00:00","proxy_end_dt":"0000-00-00 00:00:00","gl_cd":null},{uid":"101058","name":"권신혜","uspw":"6b86b273f134fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dept
00:00:00","uspw_chng_dt":"2021-12-22 00:00:00","phone_number":"01201233234","rank_cd":"","pos_cd":"지점
원","duty_cd":"1001","pfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user_type":"A","outAdminCode":"","proxy_usid":"","proxy_start_dt":
{"uid":"101111","name":"김정현B","uspw":"6b86b273f134fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dept":"23","team":"39","email":"sdfs@bisl.co.kr","role_type":
00:00:00","phone_number":"0345204234","rank_cd":"","pos_cd":"팀장","duty_cd":"1001","pfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user
00:00:00","proxy_end_dt":"0000-00-00 00:00:00","gl_cd":null},{uid":"11111111","name":"농협은행이","uspw":"6b86b273f134fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b"
00:00:00","uspw_chng_dt":"2022-01-11
00:00:00","phone_number":"","rank_cd":"RANK_N3","pos_cd":"100000201","duty_cd":"1001","pfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","us
00:00:00","proxy_end_dt":"0000-00-00 00:00:00","gl_cd":null},{uid":"161045","name":"송정하","uspw":"6b86b273f134fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dept
00:00:00","uspw_chng_dt":"2021-12-22 00:00:00","phone_number":"01042353452","rank_cd":"","pos_cd":"지점
원","duty_cd":"1001","pfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user_type":"A","outAdminCode":"","proxy_usid":"","proxy_start_dt":
{"uid":"161054","name":"박정호","uspw":"6b86b273f134fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dept":"24","team":"40","email":"chadsf@bisl.co.kr","role_type":
00:00:00","phone_number":"01051362341","rank_cd":"","pos_cd":"지점원","duty_cd":"1001","pfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","u
00:00:00","proxy_end_dt":"0000-00-00 00:00:00","gl_cd":null},{uid":"171082","name":"배담경","uspw":"6b86b273f134fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dept
00:00:00","uspw_chng_dt":"2021-12-22 00:00:00","phone_number":"01079032134","rank_cd":"","pos_cd":"지점
원","duty_cd":"1001","pfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user_type":"A","outAdminCode":"","proxy_usid":"","proxy_start_dt":
{"uid":"20011004","name":"이x승","uspw":"a4ayc/80/0Gda480/1o/V0etp0qiLxJwB5S3beHW0s=","dept":"21","team":"30","email":"xxx@bccard.com","role_type":"UA99","usid_exp_dt":"21
00:00:00","phone_number":"","rank_cd":"G10","pos_cd":"34","duty_cd":"1001","pfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user_type":
00:00:00","gl_cd":null},{uid":"20011005","name":"유x진","uspw":"a4ayc/80/0Gda480/1o/V0etp0qiLxJwB5S3beHW0s=","dept":"21","team":"30","email":"xx1@bccard.com","role_
00:00:00","phone_number":"","rank_cd":"G20","pos_cd":"34","duty_cd":"1003","pfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user_type":
00:00:00","proxy_end_dt":"0000-00-00 00:00:00","gl_cd":null},{uid":"20011008","name":"김x정","uspw":"6b86b273f134fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","de
00:00:00","uspw_chng_dt":"2021-12-22 00:00:00","phone_number":"","rank_cd":"G20","pos_cd":"24","duty_cd":"1001","pfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unusi
00:00:00 00:00:00","proxy_end_dt":"0000-00-00 00:00:00","gl_cd":null},{uid":"22222222","name":"농협은행02","uspw":"d17125de1404a2c958c3f2766f935f6331d01307daece0223643f017d
00:00:00","uspw_chng_dt":"2022-01-11
00:00:00","phone_number":"","rank_cd":"1","pos_cd":"22121","duty_cd":"3339","pfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user_type":
00:00:00","proxy_end_dt":"0000-00-00 00:00:00","gl_cd":null},{uid":"27","name":"27","uspw":"6b86b273f134fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dept":"1","t
00:00:00","uspw_chng_dt":"2022-01-11
00:00:00","phone_number":"","rank_cd":"1","pos_cd":"31122","duty_cd":"1003","pfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user_type":
00:00:00","proxy_end_dt":"0000-00-00 00:00:00","gl_cd":null},{uid":"33333333","name":"농협은행03","uspw":"469d4ca31d89de50ce129dc4b17fd66c1f58a3c1a0c3bdbb80384fd6c26c941"
00:00:00","uspw_chng_dt":"2022-01-11
00:00:00","phone_number":"","rank_cd":"","pos_cd":"100000197","duty_cd":"3334","pfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user_ty
00:00:00","proxy_end_dt":"0000-00-00 00:00:00","gl_cd":null},{uid":"abcuser","name":"r","uspw":"469d4ca31d89de50ce129dc4b17fd66c1f58a3c1a0c3bdbb80384fd6c26c941","dept":"
00:00:00","uspw_chng_dt":"2022-01-11
00:00:00","phone_number":"","rank_cd":"A_31","pos_cd":"810114","duty_cd":"3339","pfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user_ty
00:00:00","phone_number":"","rank_cd":"","pos_cd":"","duty_cd":"","pfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user_ty

```

특정한 json 타입으로 넘겨줘야 리스트가 출력됨

USID	이름	부서	팀	담당업무	이메일	전화번호
041022	박기영	24	40	1001	dfghg123@sbieob.co.kr	01023425476
10	김사강	11	18	1002		
101058	권신재	23	39	1001	sdfl@sbieob.co.kr	01201233234
101111	김정현B	23	39	1001	sdfl@sbieob.co.kr	0345204234
11111111	농협은행01	4	7	1001		
161045	송경하	26	41	1001	lermad@sbieob.co.kr	01042363452
161064	박창욱	24	40	1001	chadsl@sbieob.co.kr	01051382341
171092	배담영	25	41	1001	dammy@sbieob.co.kr	01079032134
20011004	이*순	21	30	1001	xxx@bccard.com	
20011005	유*진	21	30	1003	xx1@bccard.com	
20011008	김*정	21	30	1001	xx2@bccard.com	
22222222	농협은행02	4	7	3339		
27	27	1	1	1003	mskim@banet.co.kr	
33333333	농협은행03	4	7	3334		
abcuser	r	1	1	3339		
admin	어드민	1	1	1004	sdld	1234321
circleUser	서클유저	5	12	3333		

<https://forward.nhn.com/2020/seoul/hands-on-labs/toastui.grid-account-book/04.html>

6-6. 그외 컴퍼넌트 추가

Sidebar Btn		Adverse Event					
<input type="checkbox"/>	No.	USID	이름	부서	팀	담당업무	
<input type="checkbox"/>	1	041022	박기영	24	40	1001	dfghg1
<input type="checkbox"/>	2	10	김사강	11	18	1002	
<input type="checkbox"/>	3	101058	권신재	23	39	1001	sdfl@st
<input type="checkbox"/>	4	101111	김정현B	23	39	1001	sdfl@
<input type="checkbox"/>	5	11111111	농협은행01	4	7	1001	
<input type="checkbox"/>	6	161045	송경하	26	41	1001	lermad
<input type="checkbox"/>	7	161064	박창욱	24	40	1001	chadsl
<input type="checkbox"/>	8	171092	배담영	25	41	1001	dammy
<input type="checkbox"/>	9	20011004	이*순	21	30	1001	xxx@b
<input type="checkbox"/>	10	20011005	유*진	21	30	1003	xx1@b
<input type="checkbox"/>	11	20011008	김*정	21	30	1001	xx2@b
<input type="checkbox"/>	12	22222222	농협은행02	4	7	3339	
<input type="checkbox"/>	13	27	27	1	1	1003	mskim
<input type="checkbox"/>	14	33333333	농협은행03	4	7	3334	
<input type="checkbox"/>	15	abcuser	r	1	1	3339	
<input type="checkbox"/>	16	admin	어드민	1	1	1004	sdld

react-dom.development.js:29742
Download the React DevTools for a better development experience: <https://reactjs.org/link/react-devtools>

✖ ▶ Warning: ReactDOM.render is no longer supported in React 18. Use createRoot instead. Until you switch to the new API, your app will behave as if it's running React 17. Learn more: <https://reactjs.org/link/switch-to-createroot> react-dom.development.js:86 ⓘ

⚠ ▶ enableRowSelect has been deprecated and will be removed in a future version. Please use rowSelection instead react-data-grid.js:1 ⓘ

⚠ ▶ undefined react-data-grid.js:1 ⓘ

⚠ ▶ Warning: componentWillMount has been renamed, and is not recommended for use. See <https://reactjs.org/link/unsafe-component-lifecycles> for details. react-dom.development.js:86 ⓘ

* Move data fetching code or side effects to componentDidUpdate.

* If you're updating state whenever props change, refactor your code to use memoization techniques or move it to static getDerivedStateFromProps. Learn more at: <https://reactjs.org/link/derived-state>

* Rename componentWillMount to UNSAFE_componentWillMount to suppress this warning in non-strict mode. In React 18.x, only the UNSAFE_ name will work. To rename all deprecated lifecycles to their new names, you can run `npx react-codemod rename-unsafe-lifecycles` in your project source folder.

Please update the following components: ReactDataGrid, Viewport, t

✖ Access to fetch at 'http://localhost:8080/user/readData' from origin 'http://localhost:3000' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled. localhost:1 ⓘ

✖ Failed to load resource: net::ERR_FAILED [:8080/user/readData:1](http://localhost:8080/user/readData:1) ⚠ ⓘ

✖ Uncaught (in promise) TypeError: Failed to fetch [App.js:7](http://localhost:App.js:7) ⚠ ❌ ⓘ
at [App.js:7:1](http://localhost:App.js:7:1)
at commitHookEffectListMount (react-dom.development.js:23049:1)
at commitPassiveMountOnFiber (react-dom.development.js:24816:1)
at commitPassiveMountEffects_complete (react-dom.development.js:24781:1)
at commitPassiveMountEffects_begin (react-dom.development.js:24768:1)
at commitPassiveMountEffects (react-dom.development.js:24756:1)
at flushPassiveEffectsImpl (react-dom.development.js:26990:1)
at flushPassiveEffects (react-dom.development.js:26935:1)
at performSyncWorkOnRoot (react-dom.development.js:26032:1)
at flushSyncCallbacks (react-dom.development.js:12009:1)