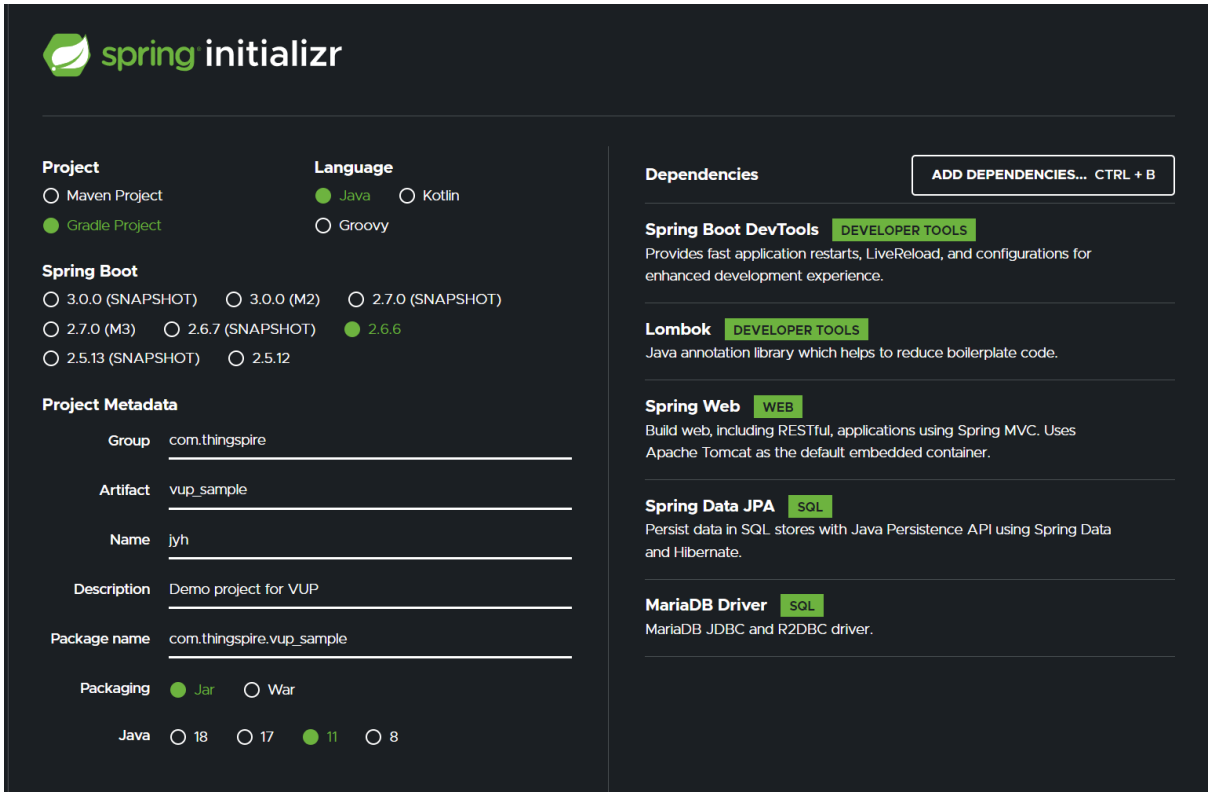
1. Spring Boot

1-1. 자바 11설치

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

1-2. 스프링부트 프로젝트 시작

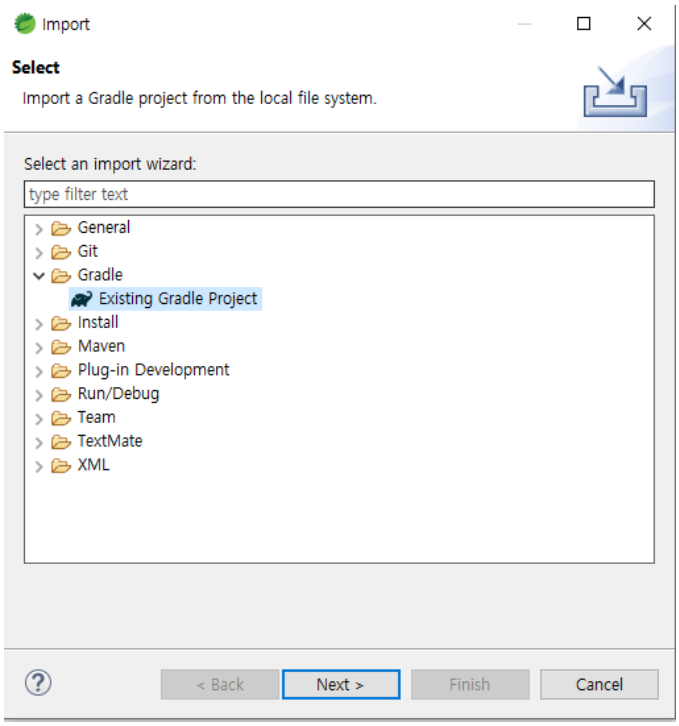
Gradle이나 Maven과 같은 빌드 도구를 이용할 수 있지만 조금 더 손쉽고 편한 방법으로 프로젝트를 만들기 위해 스프링 부트 프로젝트를 Spring initializr를 이용
https://start.spring.io



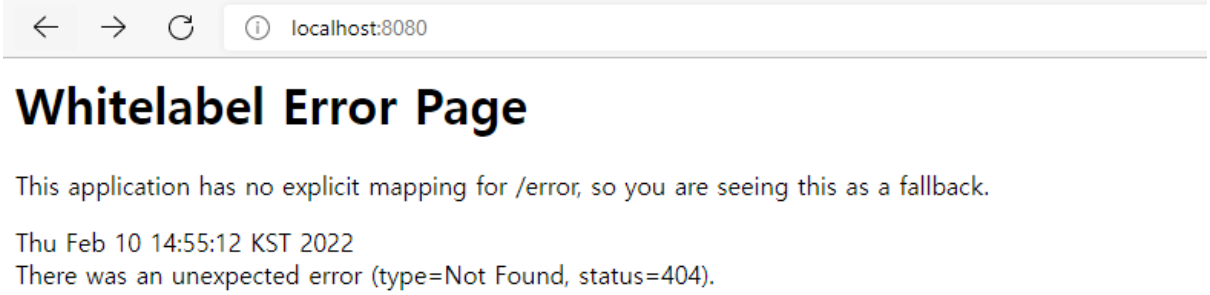
- Group: 보통 기업의 도메인 명
- Artifact: 빌드 결과물 이름, 보통 프로젝트명
- Name: 프로젝트 이름
- Description: 프로젝트 설명
- Package name: 패키지 이름, 초기 소스들이 만들어질 기본 패키지명
- Packaging: 배포 형태 (war, jar)
- Java: 자바 버전 선택
- Dependencies : 이후에 설정 파일(ex: build.gradle)에 직접 등록해도 되지만, 프로젝트 생성 전에도 라이브러리를 미리 주입할 수 있다.

1-3. 프로젝트 임포트 및 실행

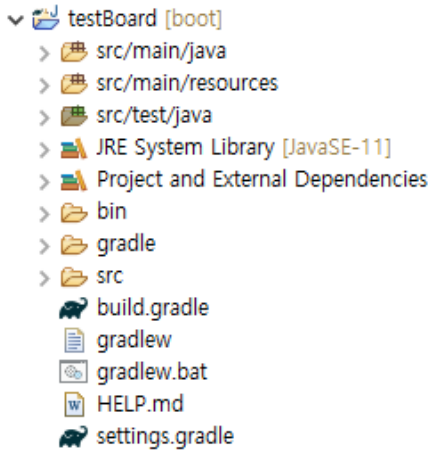
testBoard.zip

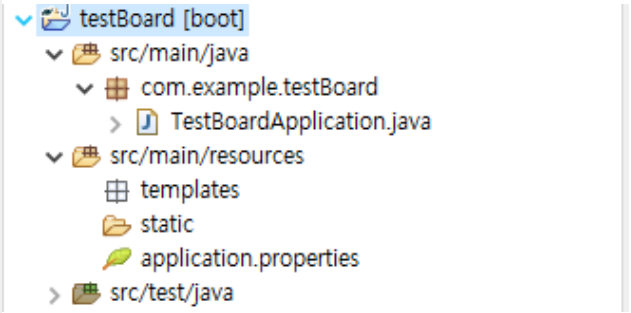



```java
package com.example.testBoard;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class TestBoardApplication {

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

}
```

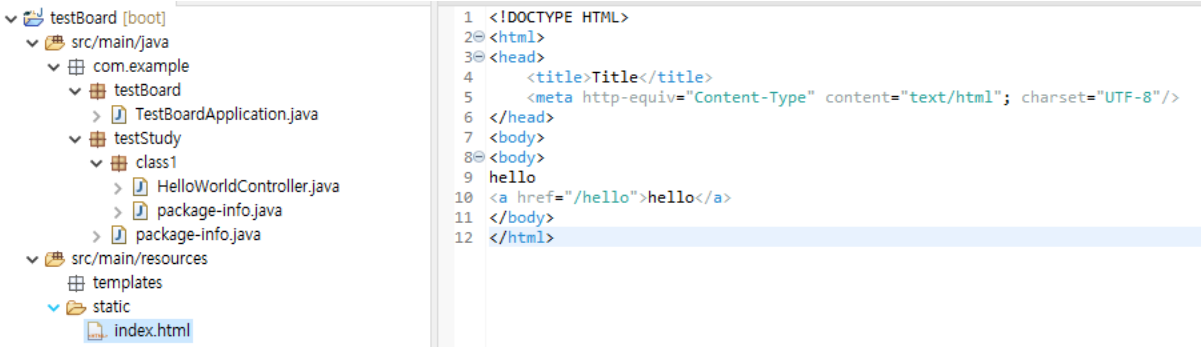


1-4. 테스트 프로젝트 구조

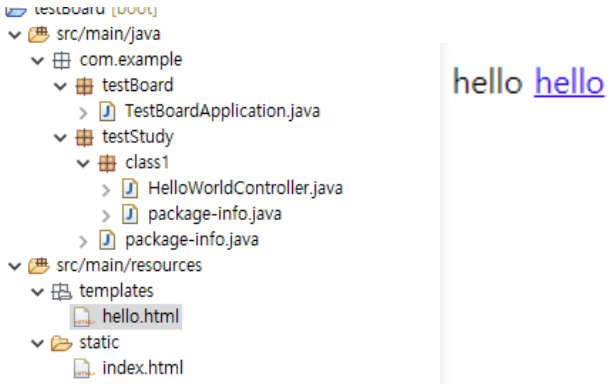




Static/index.html 웰컴페이지 기능



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-templete";
    }

    @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

```
plugins {
    id 'org.springframework.boot' version '2.6.3'
    id 'io.spring.dependency-management' version '1.0.11.RELEASE'
    id 'java'
}

group = 'com.example'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '11'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

```

helloworldController

package-info.java | HelloWorldController.java | index.html | build.gradle | hello.html

```
package com.example.testBoard.testStudy.class2;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller // 메인시작
public class HelloWorldController {
    @GetMapping("hello") // url hello
    public String hello(Model model) {
        model.addAttribute("data", "hello!!!MVC class2");
        return "hello";
    }
}

```

Templetes/hello.html

package-info.java | HelloWorldController.java | index.html | build.gradle | hello.html

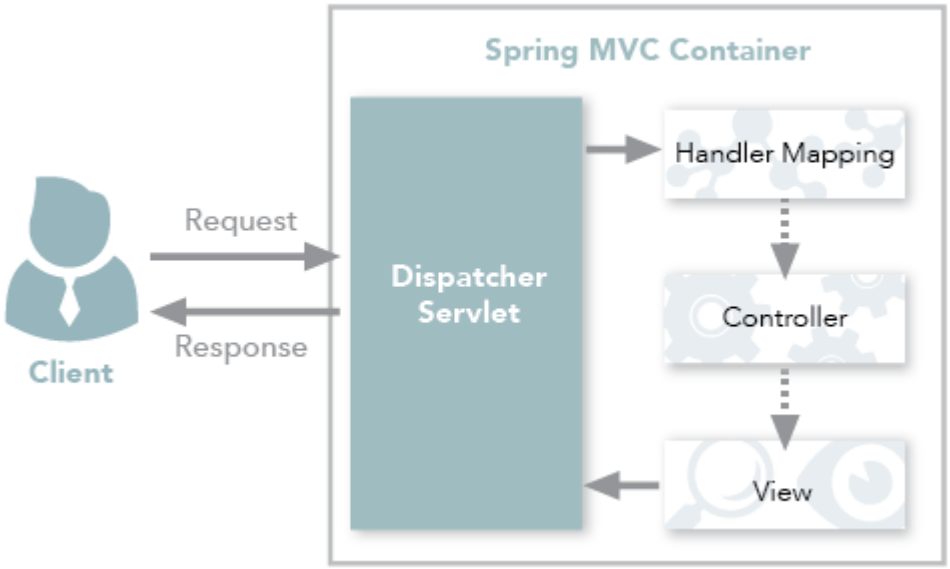
```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org"> <!--타임리프 문법 사용하겠다는 표시-->
<head>
    <title>Title</title>
    <meta http-equiv="Content-Type" content="text/html"; charset="UTF-8"/>
</head>
<body>
<p th:text="'안녕하세요. ' + ${data}">안녕하세요. 손님</p> <!--3-1의 컨트롤러에서 넘겨준 data를 받아서 화면에 처리-->
</body>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

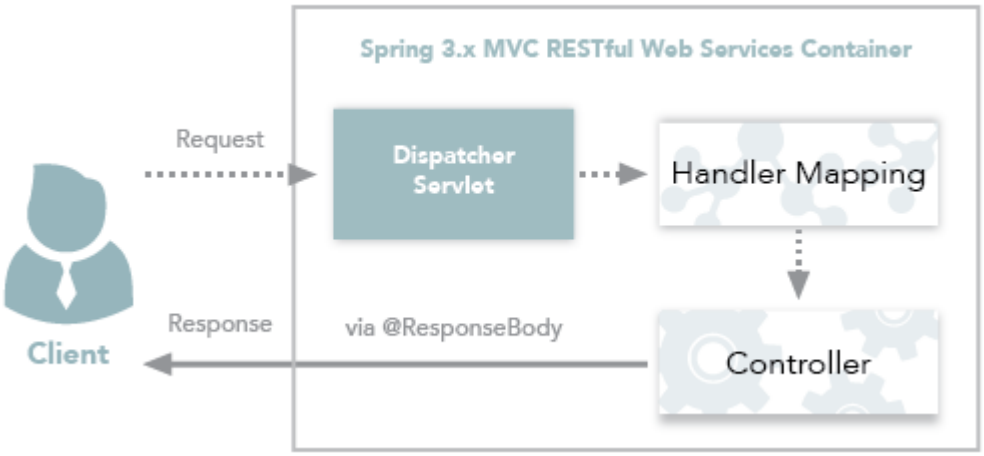
4.3 RestController 과 Controller 차이

@RestController 은 @Controller 와 @ResponseBody 의 조합입니다.

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

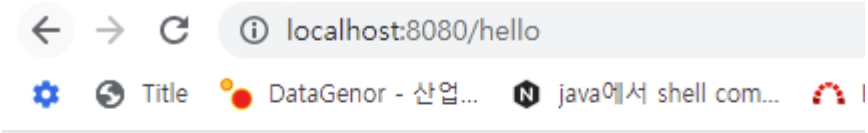
근본적인 차이점은 @Controller 의 역할은 Model 객체를 만들어 데이터를 담고 View 를 찾는 것이지만, @RestController 는 단순히 객체만을 반환하고 객체 데이터는 JSON 또는 XML 형식으로 HTTP 응답에 담아서 전송합니다. 물론 @Controller 와 @ResponseBody 를 사용하여 만들 수 있지만 이러한 방식은 RESTful 웹서비스의 기본 동작이기 때문에 Spring 은 @Controller 와 @ResponseBody 의 동작을 조합한 @RestController 을 도입했습니다.



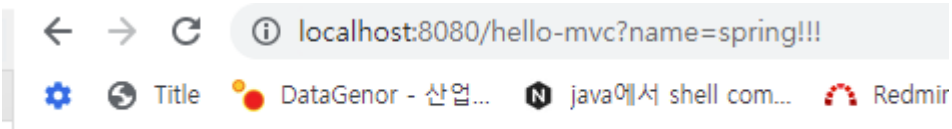


4-4. class2 – 소스참조

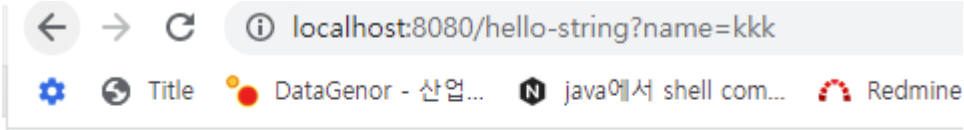
```
runtimeOnly('org.springframework.boot:spring-boot-devtools')
```



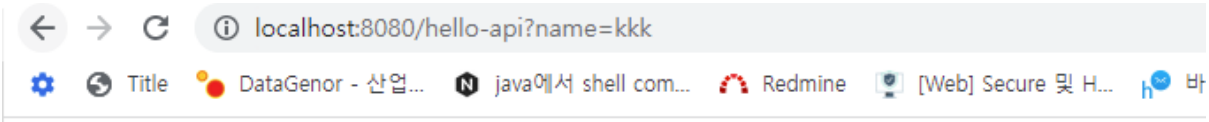
안녕하세요.hello class2!!!



안녕하세요. spring!!!

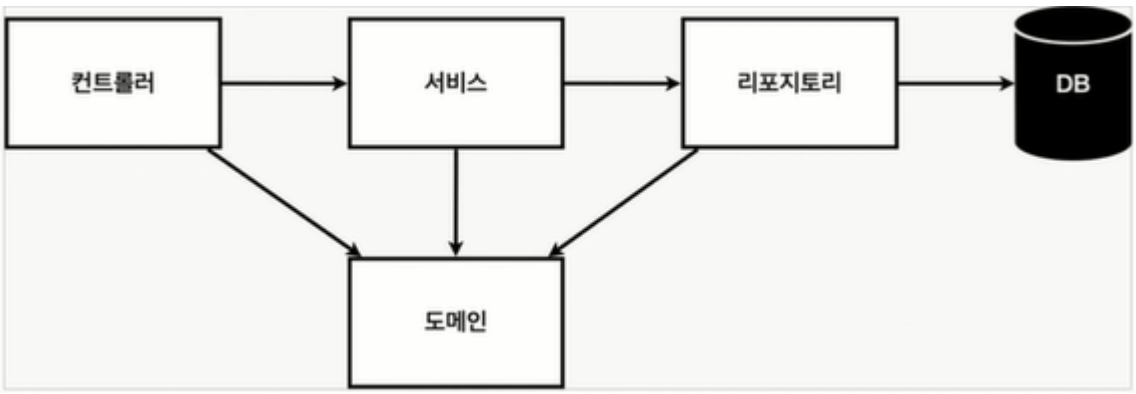


hello kkk



{"name":"kkk"}

4-5. 회원관리 예제 – 백앤드



package-info.java | Member.java

```java
package com.example.testBoard.testStudy.class3.domain;

public class Member {

    private Long id;
    private String name;

    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

package-info.java | Member.java | MemberRepository.java

```java
package com.example.testBoard.testStudy.class3.repository;

import java.util.List;

public interface MemberRepository {
    Member save(Member member);
    Optional<Member> findById(Long id);
    Optional<Member> findByName(String name);
    List<Member> findAll();
}
```

package-info.java | Member.java | MemberRepository.java | MemoryMemberRepository.java

```java
package com.example.testBoard.testStudy.class3.repository;

import java.util.ArrayList;

public class MemoryMemberRepository implements MemberRepository {

    private static Map<Long, Member> store = new HashMap<>();
    private static Long sequence = 0L;

    @Override
    public Member save(Member member) {
        member.setId(++sequence);
        store.put(member.getId(), member);
        return member;
    }

    @Override
    public Optional<Member> findById(Long id) {
        return Optional.ofNullable(store.get(id));
    }

    @Override
    public Optional<Member> findByName(String name) {
        return store.values().stream().filter(member -> member.getName().equals(name)).findAny();
    }

    @Override
    public List<Member> findAll() {
        return new ArrayList<>(store.values());
    }

    public void clearStore() {
        store.clear();
    }
}
```

package-info.java | Member.java | MemberRepository.java | MemoryMemberRepository.java | MemberService.java

```java
package com.example.testBoard.testStudy.class3.service;

import java.util.List;

public class MemberService {

    //private final MemberRepository memberRepository = new MemoryMemberRepository();
    private final MemberRepository memberRepository;

    public MemberService(MemoryMemberRepository memberRepository) {
        this.memberRepository = memberRepository;
    }

    public Long join(Member member) {
        // 같은 이름이 있는 중복회원X
        // 1
        /*
         * Optional<Member> result = memberRepository.findByName(member.getName());
         * result.ifPresent(m -> { throw new IllegalStateException("이미 존재하는 회원입니다.");
         * });
         */

        // 2
        validateDuplicateMember(member);
        memberRepository.save(member);
        return member.getId();
    }

    //2. optional을 get으로 바로 반환하는것보다 아래처럼 쓰는게 좋다.
    private void validateDuplicateMember(Member member) {
        memberRepository.findByName(member.getName()).ifPresent(m -> {
            throw new IllegalStateException("이미 존재하는 회원입니다.");
        });
    }

    public List<Member> findMembers() {
        return memberRepository.findAll();
    }

    public Optional<Member> findOne(Long memberId) {
        return memberRepository.findById(memberId);
    }
}
```

4-6. junit

package-info.java | Member.java | MemberRepository.java | MemoryMemberRepository.java | MemberService.java | MemoryMemberRepositoryTest.java

```java
public class MemoryMemberRepositoryTest {

    MemoryMemberRepository repository = new MemoryMemberRepository();

    @AfterEach
    public void afterEach() {
        repository.clearStore();
    }

    @Test
    public void save() {
        Member member = new Member();
        member.setName("Spring");
        repository.save(member);
        Member result = repository.findById(member.getId()).get();
        // System.out.println("" + (result == member));
        // Assertions.assertThat(member).isEqualTo(result);
        // Assertions.assertEquals(member, result);
        assertThat(member).isEqualTo(result);
    }

    @Test
    public void findByName() {
        Member member1 = new Member();
        member1.setName("spring1");
        repository.save(member1);

        Member member2 = new Member();
        member2.setName("spring2");
        repository.save(member2);

        Member result = repository.findByName("spring1").get();
        assertThat(result).isEqualTo(member1);
    }

    @Test
    public void findAll() {
        Member member1 = new Member();
        member1.setName("spring1");
        repository.save(member1);

        Member member2 = new Member();
        member2.setName("spring2");
        repository.save(member2);

        List<Member> result = repository.findAll();
        assertThat(result.size()).isEqualTo(2); //이전 test에 repo에 저장해서 안맞게 결과가 나옴 -> clearStore 를 해줘야 동작
    }
}
```

```java
    MemberService memberService;
    MemoryMemberRepository memberRepository;

    @BeforeEach
    public void beforeEach() {
        memberRepository = new MemoryMemberRepository();
        memberService = new MemberService(memberRepository); //di
    }

    /*
     * @AfterEach public void afterEach() { memberRepository.clearStore(); }
     */

    @Test
    void 회원가입() {
        // given
        Member member = new Member();
        member.setName("hello");

        // when
        Long svaeId = memberService.join(member);

        // then
        Member findeMember = memberService.findOne(svaeId).get();
        // Optional<findeMember> = memberService.findOne(svaeId);
        assertThat(member.getName()).isEqualTo(findeMember.getName());
    }

    @Test
    public void 중복_회원_예외() {
        // given
        Member member1 = new Member();
        member1.setName("spring1");

        Member member2 = new Member();
        member2.setName("spring1");

        // when
        memberService.join(member1);
        IllegalStateException e = assertThrows(IllegalStateException.class, () -> memberService.join(member2));
        assertThat(e.getMessage()).isEqualTo("이미 존재하는 회원입니다.");
        // assertThrows(IllegalStateException.class, () -> memberService.join(member2));
        /*
         * memberService.join(member1); try { memberService.join(member2); fail(); }
         * catch (IllegalStateException e) {
         * assertThat(e.getMessage()).isEqualTo("이미 존재하는 회원입니다."); }
         */
```

4-7. 스프링빈 의존성설정

Member.java | MemberService.java | MemberRepository.java | MemoryMemberRepository.java | package-info.java | MemberController.java

```java
package com.example.testBoard.testStudy.class4.controller;

import java.util.List;

@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 memberController을
     * 호출할때 puplic으로 되어있어야함. 따라서 변경위험성 있음)
     */
    @Autowired
    public MemberController(MemberService memberService) {
        this.memberService = memberService;
    } // DI- 생성자주입 (결론적으로 이걸 쓰는게 좋음. 의존관계가 실행중에 동적으로 변하는 경우는 거의없다)

    @GetMapping("/members/new")
    public String createForm() {
        return "members/createMemberForm";// templates에 html호출
    }

    @PostMapping("/members/new")
    public String create(MemberForm form) { // 화면에서 버튼을 누르면 name값을 받아와 메서드 실행
        Member member = new Member();
        member.setName(form.getName());

        memberService.join(member);

        return "redirect:/"; // 원래화면으로 돌아감
    }

    @GetMapping("/members")
    public String list(Model model) {
        List<Member> members = memberService.findMembers();
        model.addAttribute("members", members);
        return "members/memberList";
    }
}
```

4-8. 환경설정

MemberRepository.java | package-info.java | SpringConfig.java

```java
package com.example.testBoard.testStudy.class4;

import org.springframework.context.annotation.Bean;

@Configuration
public class SpringConfig {

    @Bean
    public MemberService memberService() {
        return new MemberService(memberRepository());
    }

    @Bean
    public MemberRepository memberRepository() {
        return new MemoryMemberRepository();
    }
}
```

4-9. DI

```java
@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 puplic으로 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

```java
package com.example.testBoard.testStudy.class4.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HomeController {

    @GetMapping("/")
    public String home() {
        return "home";
    }
}
```

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

```html
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<body>

<div class="container">
    <div>
        <h1>hello spring</h1>
        <p>회원기능</p>
        <p>
            <a href="/members/new">회원가입</a>
            <a href="/members">회원목록</a>
        </p>
    </div>
</div>
</body>
</html>
```

---

hello spring

회원기능

회원가입 회원목록

4-10. 메모리 기본 cr

MemberRepository.java | package-info.java | **createMemberForm.html** | memberList.html

```html
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<body>

<div class="container">
    <form action="/members/new" method="post">
        <div class="form-group">
            <label for="name">이름</label>
            <input type="text" id="name" name="name" placeholder="이름을 입력하세요">
        </div>
        <button type="submit">등록</button>
    </form>
</div>
</body>
</html>
```

MemberRepository.java | package-info.java | createMemberForm.html | **memberList.html**

```html
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<body>

<div class="container">
    <div>
        <table>
            <thead>
            <tr>
                <th>#</th>
                <th>이름</th>
            </tr>
            </thead>
            <tbody>
            <tr th:each="member : ${members}">
                <td th:text="${member.id}"></td>
                <td th:text="${member.name}"></td>
            </tr>
            </tbody>
        </table>
    </div>
</div>
</body>
</html>
```

localhost:8080/members/new

이름 sdf

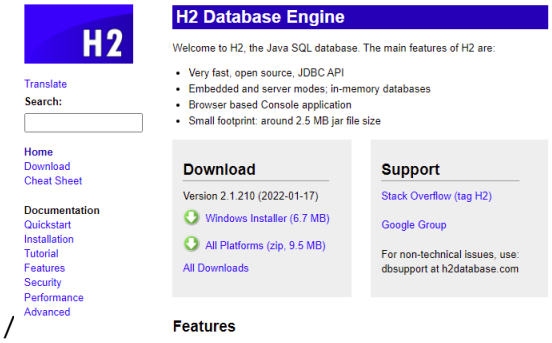
등록

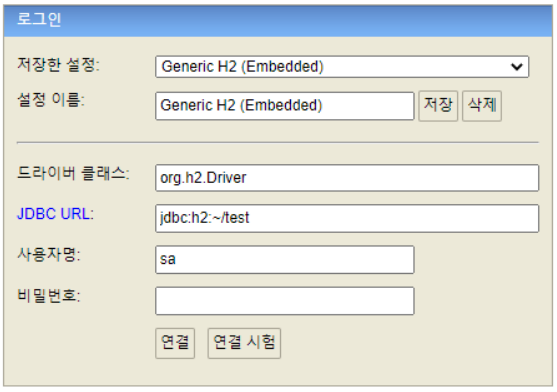
localhost:8080/members

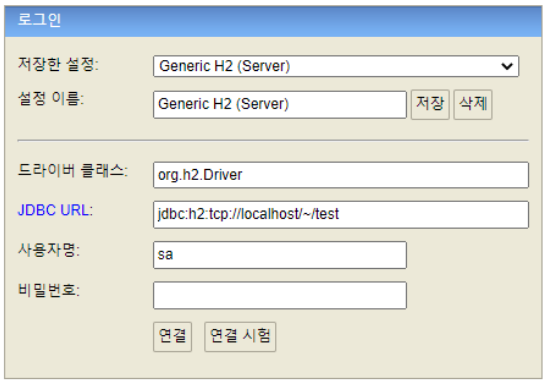
#	이름
1	123123
2	123123435
3	dfgdgvf
4	324324

4-11. 디비 연동 – h2 db

https://www.h2database.com







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...

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

```java
package com.example.testBoard.testStudy.class5.domain;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Member {

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    //@Column(name="username")
    private String name;

    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

4-12. Repository교체

4-12-1. Jdbc

package-info.java | JdbcMemberRepository.java

```java
package com.example.testBoard.testStudy.class5.repository;

import java.sql.Connection;

//귀찮 대충 하나만
public class JdbcMemberRepository implements MemberRepository {

    private final DataSource datasource;

    public JdbcMemberRepository(DataSource datasource) {
        this.datasource = datasource;
    }

    private Connection getConnection() {
        return  DataSourceUtils.getConnection(datasource);//이걸 써야 커넥션이 여러개 안생김
    }

    @Override
    public Member save(Member member) {
        String sql = "insert into member(name) values(?)";

        Connection conn = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;

        try {
            conn = getConnection();
            pstmt = conn.prepareStatement(sql, java.sql.Statement.RETURN_GENERATED_KEYS);

            pstmt.setString(1, member.getName());
            pstmt.executeUpdate();

            rs = pstmt.getGeneratedKeys();

            if (rs.next()) {
                member.setId(rs.getLong(1));
            } else {
                throw new SQLException("id 조회실패");
```

4-12-2. JdbcTemplate

package-info.java | JdbcMemberRepository.java | JdbcTemplateMemberRepository.java

```java
package com.example.testBoard.testStudy.class5.repository;

import java.util.HashMap;

public class JdbcTemplateMemberRepository implements MemberRepository {

    private final JdbcTemplate jdbcTemplate;

    @Autowired // 생성자 하나면 autowired생략가능
    public JdbcTemplateMemberRepository(DataSource dataSource) {
        jdbcTemplate = new JdbcTemplate(dataSource);
    }

    @Override
    public Member save(Member member) {
        SimpleJdbcInsert jdbcInsert = new SimpleJdbcInsert(jdbcTemplate);
        jdbcInsert.withTableName("member").usingGeneratedKeyColumns("id");

        Map<String, Object> parameters = new HashMap<>();
        parameters.put("name", member.getName());

        Number key = jdbcInsert.executeAndReturnKey(new MapSqlParameterSource(parameters));
        member.setId(key.longValue());
        return member;
    }

    @Override
    public Optional<Member> findById(Long id) {
        List<Member> result = jdbcTemplate.query("select * from member where id = ?", memberRowMapper(), id);
        return result.stream().findAny();
    }

    @Override
    public Optional<Member> findByName(String name) {
        List<Member> result = jdbcTemplate.query("select * from member where name = ?", memberRowMapper(), name);
        return result.stream().findAny();
    }
```

4-12-3. Jpa

package-info.java | JdbcMemberRepository.java | JdbcTemplateMemberRepository.java | JpaMemberRepository.java

```java
package com.example.testBoard.testStudy.class5.repository;

import java.util.List;

public class JpaMemberRepository implements MemberRepository {

    private EntityManager em;// 이게 데이터소스 커넥션등 다 들고있음

    public JpaMemberRepository(EntityManager em) {
        this.em = em;
    }

    @Override
    public Member save(Member member) {
        em.persist(member);
        return member;
    }

    @Override
    public Optional<Member> findById(Long id) {
        Member member = em.find(Member.class, id);
        return Optional.ofNullable(member);
    }

    @Override
    public Optional<Member> findByName(String name) {
        List<Member> result = em.createQuery("select m from Member m where m.name = :name", Member.class)
                .setParameter("name", name).getResultList();
        return result.stream().findAny();
    }

    @Override
    public List<Member> findAll() {
        return em.createQuery("select m from Member m", Member.class).getResultList();
    }
}
```

4-12-4. SpringDataJpa

package-info.java | JdbcMemberRepository.java | JdbcTemplateMemberRepository.java | JpaMemberRepository.java | SpringDataJpaMemberRepository.java

```java
package com.example.testBoard.testStudy.class5.repository;

import java.util.Optional;

public interface SpringDataJpaMemberRepository extends JpaRepository<Member, Long>, MemberRepository {

    @Override
    Optional<Member> findByName(String name);
}
```

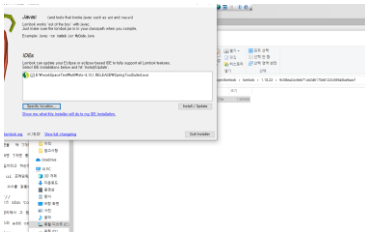
4-13. Aop

package-info... | JdbcMemberRe... | JdbcTemplate... | JpaMemberRep... | SpringDataJp... | TimeTraceAop... | HomeControll... | SpringConfig... | TestBoardApp...

```java
package com.example.testBoard.testStudy.class5.aop;

import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class TimeTraceAop {

    @Around("execution(* com.example.testBoard..*(..))")
    public Object execute(ProceedingJoinPoint joinPoint) throws Throwable {
        long start = System.currentTimeMillis();
        System.out.println("start : " + joinPoint.toString());
        try {
            return joinPoint.proceed();
        } finally {
            long finish = System.currentTimeMillis();
            long timeMs = finish - start;
            System.out.println("end : " + joinPoint.toString());
            System.out.println("ms : " + timeMs);
        }
    }
}
```

```
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... | JdbcMemberR... | JdbcTemplat... | JpaMemberRe... | SpringConfi... | TestBoardAp... | Member.java

```java
package com.example.testBoard.testStudy.class5.domain;

import javax.persistence.Column;

@Entity
@Data
public class Member {

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    //@Column(name="username")
    private String name;
/*
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
*/
}
```

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=updae
# 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/[redacted]?useSSL=false&useUnicode=true&serverTimezone=Asia/Seoul
# DB username
spring.datasource.username=[redacted]
# DB password
spring.datasource.password=[redacted]
```

6-2. 롬복 및 스프링데이터 jpa

```
2022-02-23 14:28:02.904  INFO 15512 --- [  restartedMain] org.hibernate.d
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;
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

```java
@Transactional
public class UserServiceTest {
    @Autowired
    UserRepository userRepository;

    @Test
    void 사용자추가() {
        User user = new User();
        user.setUsid("testUser01");
        user.setName("테스트유저01");
        user.setDept("1");
        user.setTeam("1");
        user.setLocked("N");
        user.setPwfailcnt(0);

        User saveUser = userRepository.save(user);
        assertThat(user.getName()).isEqualTo(saveUser.getName());
    }

    @Test
    void 사용자조회() {
        User user = new User();
        user.setUsid("admin");
        user.setName("어드민");

        Optional<User> findUser = userRepository.findById(user.getUsid());
        //System.out.println(findUser.toString());
        assertThat(user.getName()).isEqualTo(findUser.get().getName());

        List<User> userList = userRepository.findAll();
        userList.stream().forEach(s -> System.out.println(s));

```

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

```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)]

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////

```java
@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

Title DataGenor - 산업... java에서 shell com... Redmine [Web] Secure 및 H... 바넷정보기술 애드몹(AdMob) 앱... APK 분석하기 위한... [상태크 따라하기①...

[{"usid":"041022","name":"박기영","uspw":"6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dept":"24","team":"40","email":"dfghfg1230@sbisb.co.kr","role_typ
00:00:00","phone_number":"01023425476","rank_c":"","pos_c":"지점장","duty_c":"1001","pwfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","
00:00:00","proxy_end_dt":"0000-00-00 00:00:00","gl_cd":null},{"usid":"10","name":"김사장","uspw":"6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dept":"
00:00:00","uspw_chng_dt":"2099-07-28
00:00:00","phone_number":"","rank_c":"A_21","pos_c":"A_201","duty_c":"1002","pwfailcnt":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},{"usid":"101058","name":"권신혜","uspw":"6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dept
00:00:00","uspw_chng_dt":"2021-12-22 00:00:00","phone_number":"01201233234","rank_c":"","pos_c":"지점
원","duty_c":"1001","pwfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user_type":"A","outAdminCode":"","proxy_usid":"","proxy_start_dt
{"usid":"101111","name":"김정현B","uspw":"6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dept":"23","team":"39","email":"sdfsf@sbisb.co.kr","role_type":
00:00:00","phone_number":"0345204234","rank_c":"","pos_c":"팀장","duty_c":"1001","pwfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","use
00:00:00","proxy_end_dt":"0000-00-00 00:00:00","gl_cd":null},{"usid":"11111111","name":"농협은행01","uspw":"6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b
00:00:00","uspw_chng_dt":"2022-01-11
00:00:00","phone_number":"","rank_c":"RANK_N3","pos_c":"100000201","duty_c":"1001","pwfailcnt":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},{"usid":"161045","name":"송정하","uspw":"6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dept
00:00:00","uspw_chng_dt":"2021-12-22 00:00:00","phone_number":"01042353452","rank_c":"","pos_c":"지점
원","duty_c":"1001","pwfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user_type":"A","outAdminCode":"","proxy_usid":"","proxy_start_dt
{"usid":"161064","name":"박창욱","uspw":"6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dept":"24","team":"40","email":"chadsf@sbisb.co.kr","role_type":
00:00:00","phone_number":"01051382341","rank_c":"","pos_c":"지점원","duty_c":"1001","pwfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","
00:00:00","proxy_end_dt":"0000-00-00 00:00:00","gl_cd":null},{"usid":"171092","name":"배담영","uspw":"6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dept
00:00:00","uspw_chng_dt":"2021-12-22 00:00:00","phone_number":"01079032134","rank_c":"","pos_c":"지점
원","duty_c":"1001","pwfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user_type":"A","outAdminCode":"","proxy_usid":"","proxy_start_dt
{"usid":"20011004","name":"이x승","uspw":"a4ayc/80/OGda4B0/lo/VOetpOqiLx1JwB5S3beHWOs=","dept":"21","team":"30","email":"xxx@bccard.com","role_type":"UA99","usid_exp_dt":"20
00:00:00","phone_number":"","rank_c":"G10","pos_c":"34","duty_c":"1001","pwfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user_type":"
00-00 00:00:00","gl_cd":null},{"usid":"20011005","name":"유x진","uspw":"a4ayc/80/OGda4B0/lo/VOetpOqiLx1JwB5S3beHWOs=","dept":"21","team":"30","email":"xx1@bccard.com","role.
00:00:00","phone_number":"","rank_c":"G20","pos_c":"34","duty_c":"1003","pwfailcnt":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},{"usid":"20011008","name":"김x점","uspw":"6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dep
00:00:00","uspw_chng_dt":"2021-12-22 00:00:00","phone_number":"","rank_c":"G20","pos_c":"24","duty_c":"1001","pwfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unuse
00-00 00:00:00","proxy_end_dt":"0000-00-00 00:00:00","gl_cd":null},{"usid":"22222222","name":"농협은행02","uspw":"d17125de1404a2c958c3f2766f93516331d01307daece0223e643f017d
00:00:00","uspw_chng_dt":"2022-01-11
00:00:00","phone_number":"","rank_c":"1","pos_c":"22121","duty_c":"3339","pwfailcnt":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},{"usid":"27","name":"27","uspw":"6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b","dept":"1","
00:00:00","uspw_chng_dt":"2022-01-11
00:00:00","phone_number":"","rank_c":"1","pos_c":"31122","duty_c":"1003","pwfailcnt":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},{"usid":"33333333","name":"농협은행03","uspw":"469d4ca31d89de50ce129dc4bf7dfd66c1f58a3c1a0c3bdbb80384fd6c26c941
00:00:00","uspw_chng_dt":"2022-01-11
00:00:00","phone_number":"","rank_c":"1","pos_c":"100000197","duty_c":"3334","pwfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"","otp_use_yn":"N","user_typ
00:00:00","proxy_end_dt":"0000-00-00 00:00:00","gl_cd":null},{"usid":"abcuser","name":"r","uspw":"469d4ca31d89de50ce129dc4bf7dfd66c1f58a3c1a0c3bdbb80384fd6c26c941","dept":"
00:00:00","uspw_chng_dt":"2022-01-11
00:00:00","phone_number":"","rank_c":"A_31","pos_c":"810114","duty_c":"3339","pwfailcnt":0,"locked":"N","outAdminUsid":"","user_ip":"","unused":"N","otp_use_yn":"N","user_ty

특정한 json 타입으로 넘겨줘야 리스트가 출력됨

USID	이름	부서	팀	담당업무	이메일	전화번호
041022	박기영	24	40	1001	dfghfg123@sbisb.co.kr	01023425476
10	김사장	11	18	1002		
101058	권신혜	23	39	1001	sdf@sbisb.co.kr	01201233234
101111	김정현B	23	39	1001	sdfsf@sbisb.co.kr	0345204234
11111111	농협은행01	4	7	1001		
161045	송정하	26	41	1001	lermad@sbisb.co.kr	01042353452
161064	박창욱	24	40	1001	chadsf@sbisb.co.kr	01051382341
171092	배담영	25	41	1001	dammy@sbisb.co.kr	01079032134
20011004	이x승	21	30	1001	xxx@bccard.com	
20011005	유x진	21	30	1003	xx1@bccard.com	
20011008	김x정	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	sdfd	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

```
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: componentWillReceiveProps 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 componentWillReceiveProps to UNSAFE_componentWillReceiveProps 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
Uncaught (in promise) TypeError: Failed to fetch   App.js:7
    at 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)
```