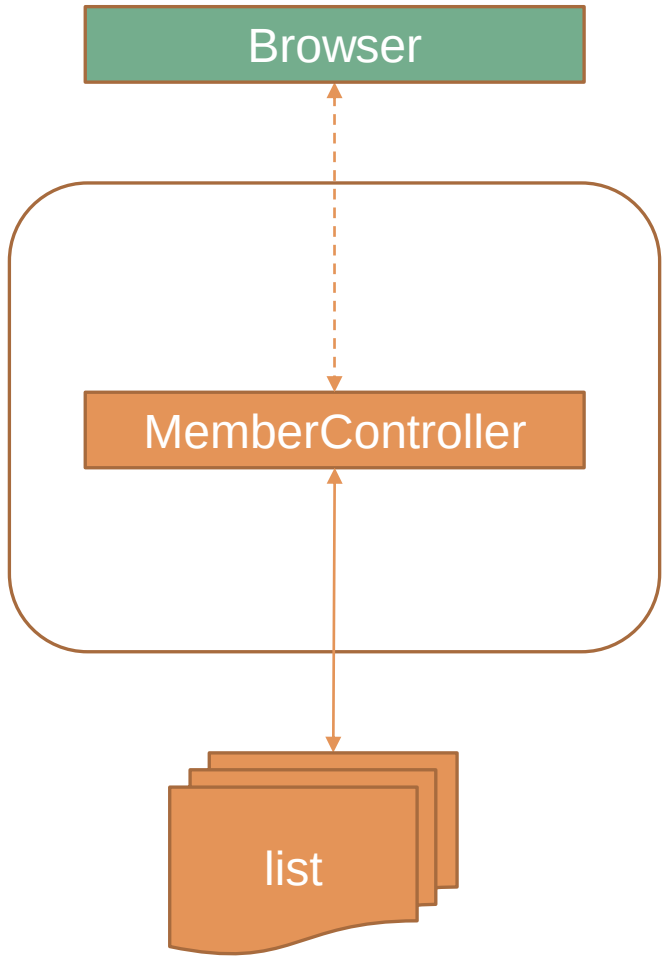
SpringBoot Missions

이상현

Mission 1 – List

- 구조
  - edu.pnu.controller
    - MemberController : 웹브라우저를 통한 요청을 받아들이고 응답하는 컨트롤러 객체
    - 클래스의 속성으로 List<MemberVO> 타입의 참조 변수를 선언, 생성자에서 메모리를 할당하여 MemberVO 객체 10개를 생성해서 저장
  - edu.pnu.domain
    - MemberVO.java : 데이터 객체




```
public class MemberVO {
    private Integer id;
    private String pass;
    private String name;
    private Date regidate;
}
```

Mission 1 – API List Example

- [GET] http://localhost:8080/members
  - 모든 멤버 정보를 JSON 형태로 브라우저에 출력
- [GET] http://localhost:8080/member?id=1
  - 아이디가 1 인 member를 찾아서 브라우저에 출력
- [POST] http://localhost:8080/member
  - member 정보를 파라미터로 전달하고 추가된 객체를 출력
- [PUT] http://localhost:8080/member
  - 수정 객체 정보를 파라미터로 전달하고 수정된 객체를 출력
- [DELETE] http://localhost:8080/member?id=5
  - 아이디가 5 인 member를 찾아서 삭제, 브라우저에는 삭제된 객체 수를 출력

C : Create	- Post
R : Read	- Get
U : Update	- Put
D : Delete	- Delete

```java
@RestController
public class MemberController {

    private List<MemberVO> list = new ArrayList<>();

    public MemberController() {
        for (int i = 1 ; i <= 10 ; i++ ) {
            list.add(MemberVO.builder()
                .id(i).name("name" + i)
                .pass("pass" + i)
                .regidate(new Date()).build());
        }
    }

    // 검색(Read - select)
    @GetMapping("/members")
    public List<MemberVO> getAllMember() {
        return list;
    }

    // 검색(Read - select)
    @GetMapping("/member")
    public MemberVO getMemberById(Integer id) {
        for (MemberVO m : list) {
            if (m.getId() == id)
                return m;
        }
        return null;
    }

    // 입력(Create - insert)
    @PostMapping("/member")
    public MemberVO addMember(MemberVO memberVO) {
        if (getMemberById(memberVO.getId()) != null) {
            System.out.println(memberVO.getId() +
                                        "가 이미 존재합니다.");
            return null;
        }
        memberVO.setRegidate(new Date());
        list.add(memberVO);
        return memberVO;
    }

    // 수정(Update - update)
    @PutMapping("/member")
    public int updateMember(MemberVO memberVO) {
        MemberVO m = getMemberById(memberVO.getId());
        if (m == null)
            return 0;
        m.setName(memberVO.getName());
        m.setPass(memberVO.getPass());
        return 1;
    }

    // 삭제(Delete - delete)
    @DeleteMapping("/member")
    public int removeMember(Integer id) {
        try {
            list.remove(getMemberById(id));
        } catch(Exception e) {
            return 0;
        }
        return 1;
    }
}
```

Mission 1 – API List Example

- [POST] http://localhost:8080/memberJSON
  - member 정보를 Request객체의 Body에 전달하고 추가된 객체를 출력

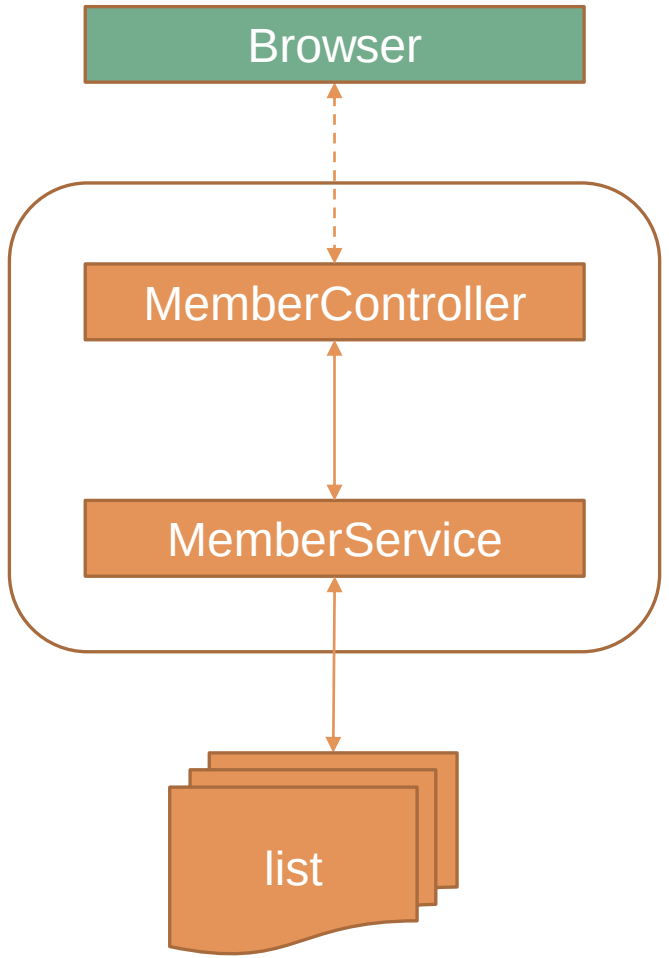
```
// 입력(Create - insert)
@PostMapping("/memberJSON")
public MemberVO addMemberJSON(@RequestBody MemberVO memberVO) {
    if (getMemberById(memberVO.getId()) != null) {
        System.out.println(memberVO.getId() + "가 이미 존재합니다.");
        return null;
    }
    memberVO.setRegidate(new Date());
    list.add(memberVO);
    return memberVO;
}
```

```
@PostMapping("/memberJSON")
public MemberVO addMemberJSON(@RequestBody MemberVO memberVO) {
    return addMember(memberVO);
}
```

Mission 2 – List

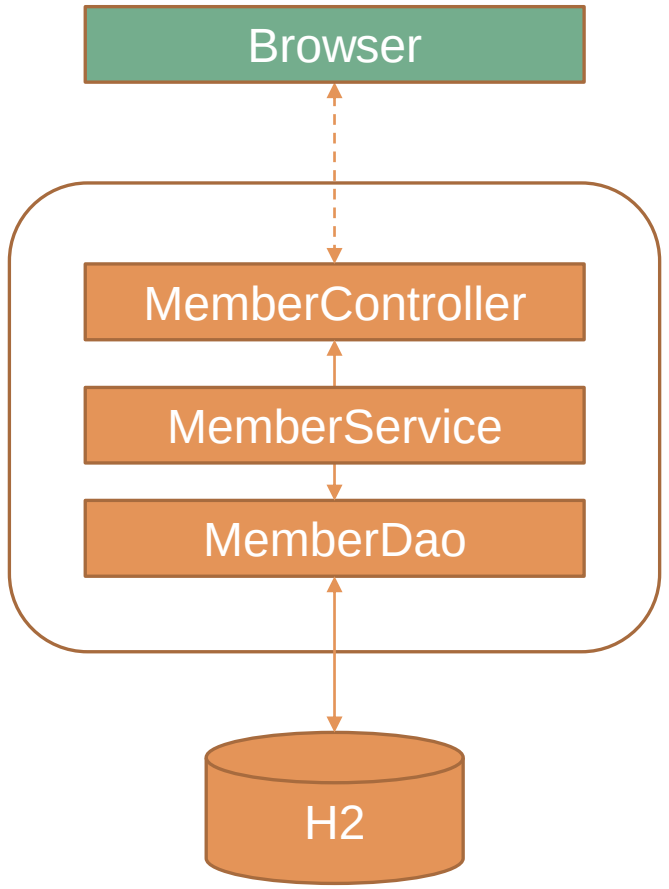
- 구조
  - edu.pnu.controller
    - MemberController : 웹브라우저를 통한 요청을 받아들이는 컨트롤러 객체
    - MemberService 참조 변수를 선언하고 생성자에서 객체 생성
  - edu.pnu.service
    - MemberService : 컨트롤러에게 데이터를 제공하는 서비스 객체
    - Mission1에서 컨트롤러에 정의했던 List<MemberVO> 변수를 여기에서 선언
  - edu.pnu.domain
    - MemberVO.java : 데이터 객체

```
public class MemberVO {
    private Integer id;
    private String pass;
    private String name;
    private Date regidate;
}
```



Mission 3 – DataBase

- 내용
  - Mission2의 MemberService에서 데이터를 제공하는 방식에서 MemberDao를 만들어서 데이터베이스에서 제공하는 방식으로 업그레이드
  - 테스트를 위한 데이터베이스는 H2 Database를 사용
- 구조
  - edu.pnu.controller
    - MemberController : 웹브라우저를 통한 요청을 받아들이는 컨트롤러 객체
  - edu.pnu.service
    - MemberService : 컨트롤러에게 데이터를 제공하는 객체
  - edu.pnu.dao
    - MemberDao : 데이터베이스 접근 객체
  - edu.pnu.domain
    - MemberDTO : 데이터 객체

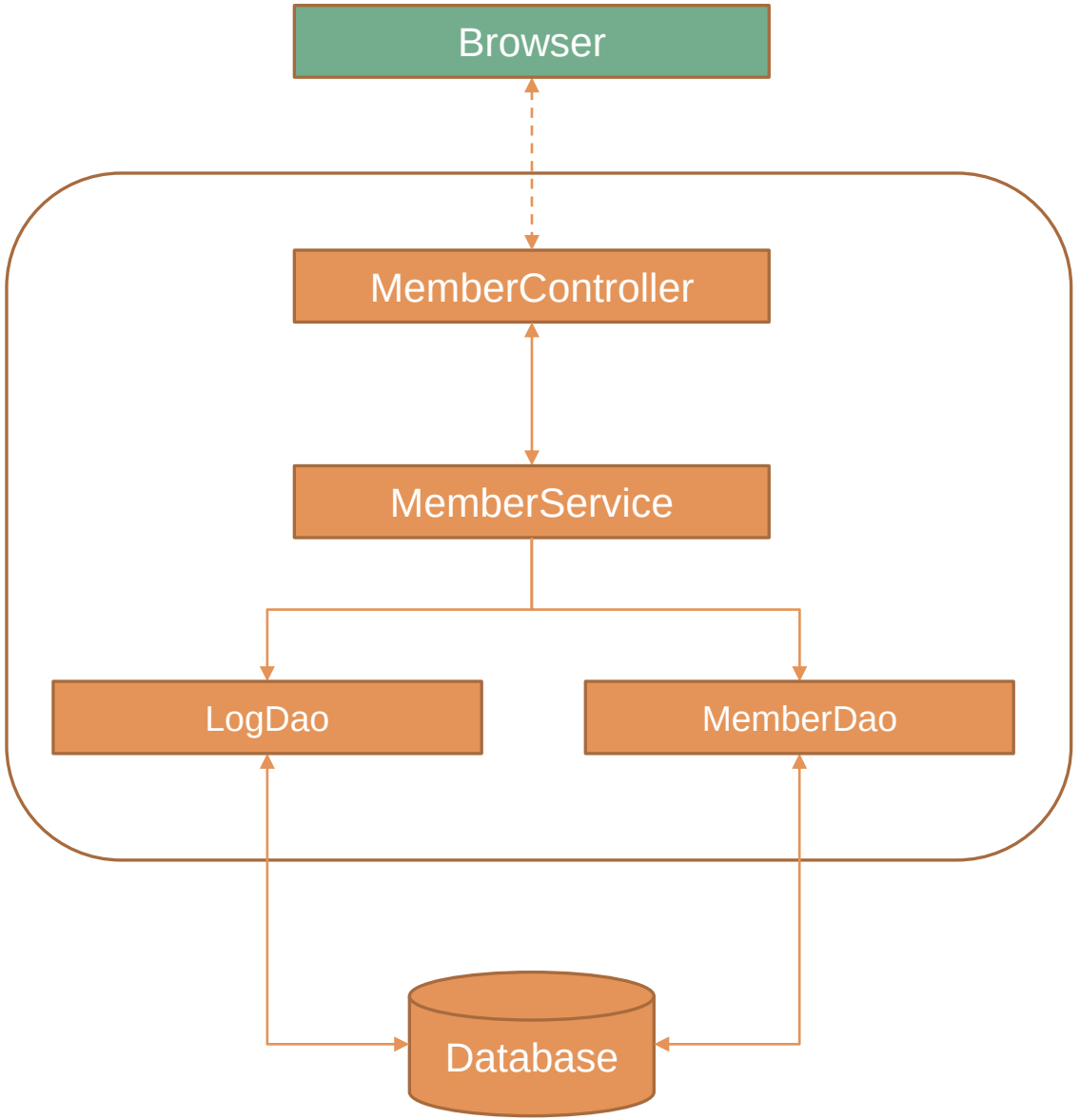


```
create table member (
        id int auto_increment primary key,
        pass varchar(10) not null,
        name varchar(20) not null,
        regidate date default (curdate()) not null
)
insert into member (pass, name) values (‘pass1’, ‘name1’);
insert into member (pass, name) values (‘pass2’, ‘name2’);
Insert into member (pass, name) values (‘pass3’, ‘name3’);
Insert into member (pass, name) values (‘pass4’, ‘name4’);
Insert into member (pass, name) values (‘pass5’, ‘name5’);
```

Mission 4 – DataBase

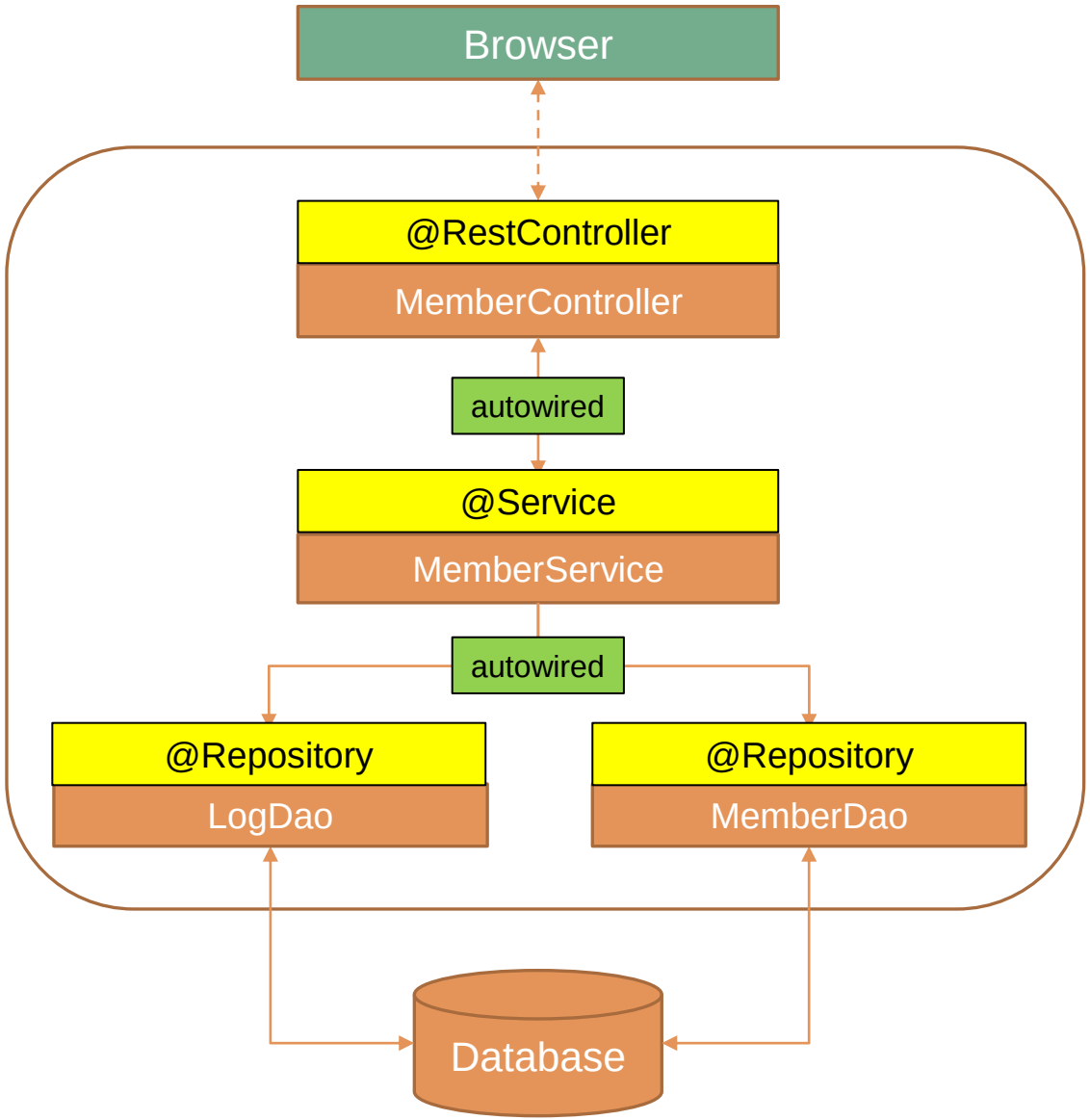
- 내용
  - Log를 남기는 기능 추가
- 구조
  - edu.pnu.dao
    - LogDao.java : 로그 데이터베이스 접근 객체

```
create table dblog (
        id int auto_increment primary key,
        method varchar(10) not null,
        sqlstring varchar(256) not null, ➔ optional
        regidate date default (curdate()) not null,
        success boolean default true
)
```



Mission 5 – DI

- 내용
  - Mission4의 모든 객체들의 연결을 Annotation을 이용한 DI로 변경
  - DI Annotation
    - @Controller
    - @RestController
    - @Service
    - @Repository
    - @Configuration - @Bean
    - @Component

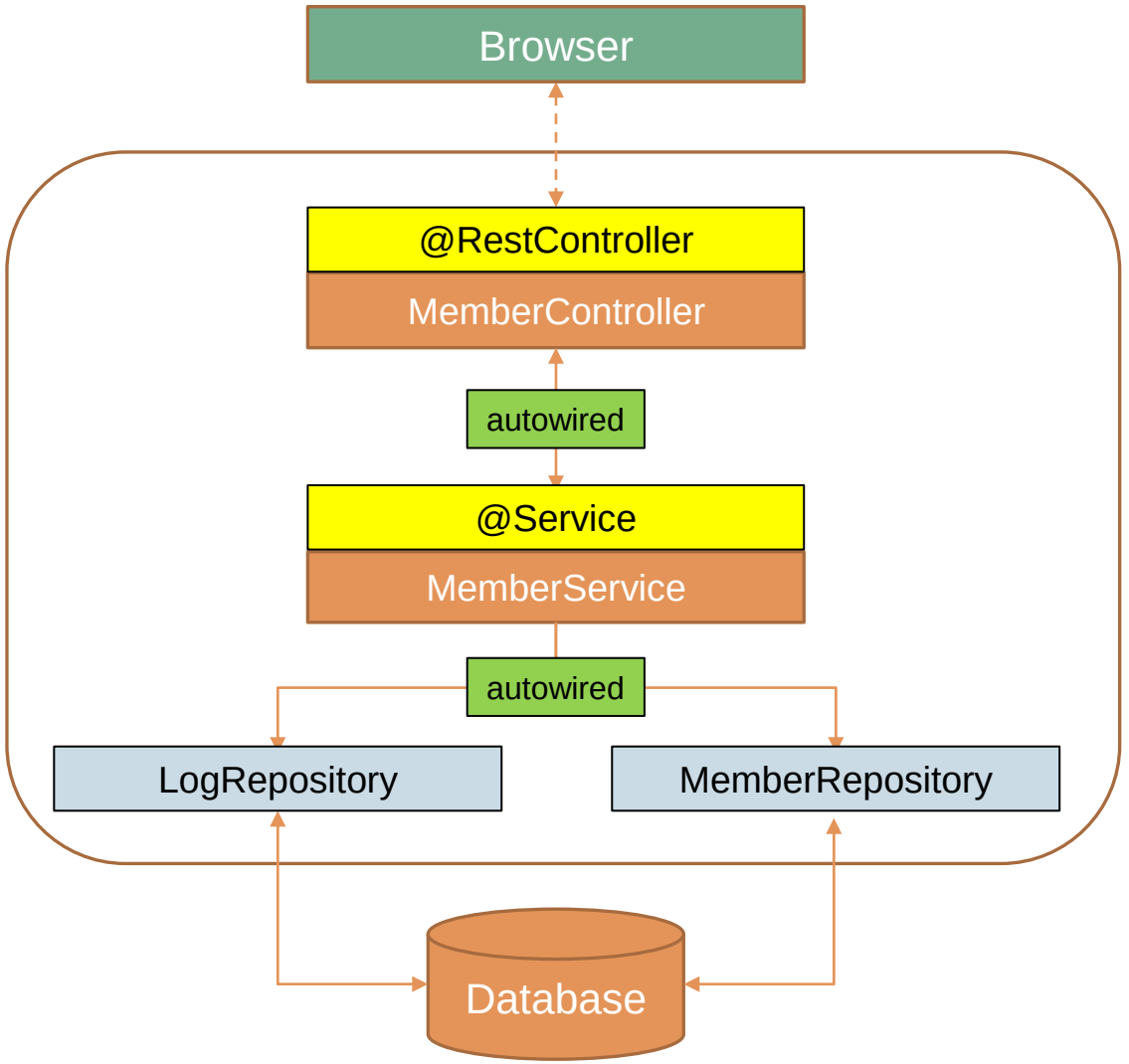


Mission 5

- DI 방법
  1. 필드에 Autowired를 적용하는 방법
     - @Autowired
     - private MemberService memberService;
  2. 생성자를 이용하는 방법
     - @ Autowired
     - public MemberController(MemberService memberService)
  3. Setter를 이용한 방법
     - @ Autowired
     - public void setMemberService(MemberService memberService)
  4. Lombok Annotation을 이용한 방법
     - @RequiredArgsConstructor
     - 필드 앞에 final 예약어가 필요

Mission 6 – JPA

- 내용
  - Mission5의 Dao 클래스를 JPA 인터페이스로 변경
- 구조
  - edu.pnu.controller
    - MemberController : 요청 컨트롤러 객체
  - edu.pnu.service
    - MemberService : 컨트롤러에게 데이터를 제공하는 객체
  - edu.pnu.persistence
    - MemberRepository : 데이터베이스 접근 인터페이스
    - LogRepository : 데이터베이스 접근 인터페이스

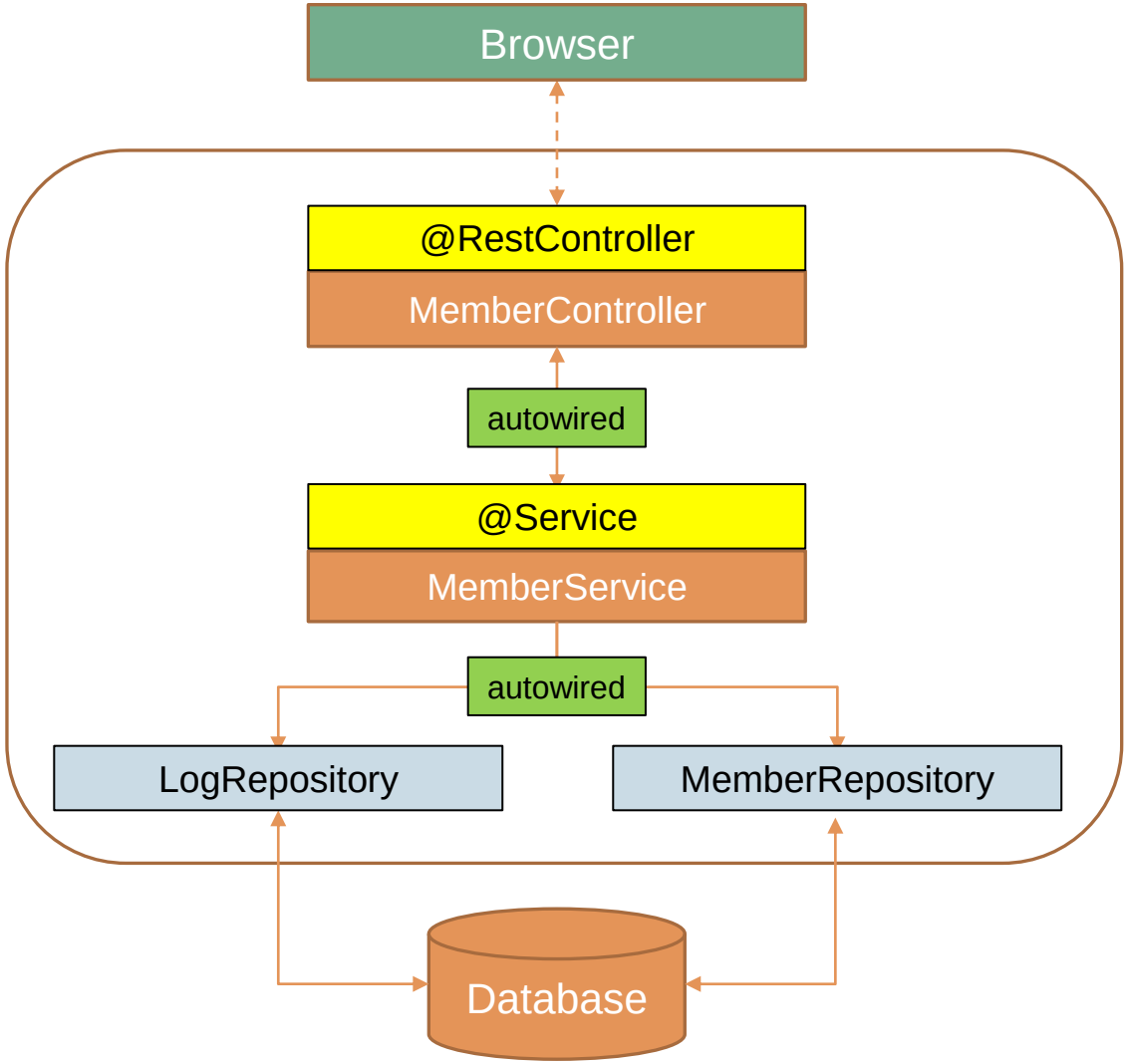


Mission 6 – JPA 프로젝트 진행 순서

1. Entity 클래스 작성 (Member) - @Entity
2. Repository Interface 작성 (MemberRepository)
   - @Repository를 설정할 클래스를 Boot가 자동으로 생성해서 처리함.
3. application.properties 설정
4. 테스트 코드를 작성해서 Repository가 정상 작동하는지 확인. (RepositoryTest)
5. Service 작성 (MemberService) - @Service
   - Repository 호출
6. 테스트 코드를 작성해서 Service가 정상 작동하는지 확인. (ServiceTest)
7. Controller 작성 (MemberController) - @RestController
   - URL 매핑 & Service 호출

Mission 7 – Security

- 내용
  - Mission6에 JWT를 이용한 인증/인가 기능 추가
- 구조
  - edu.pnu.config
    - SecurityConfig
  - edu.pnu.config.auth
    - JWTAuthenticationFilter (인증)
    - JWTAuthorizationFilter (인가)
- 권한
  - Get Method – all
  - Post/Put/Delete - User

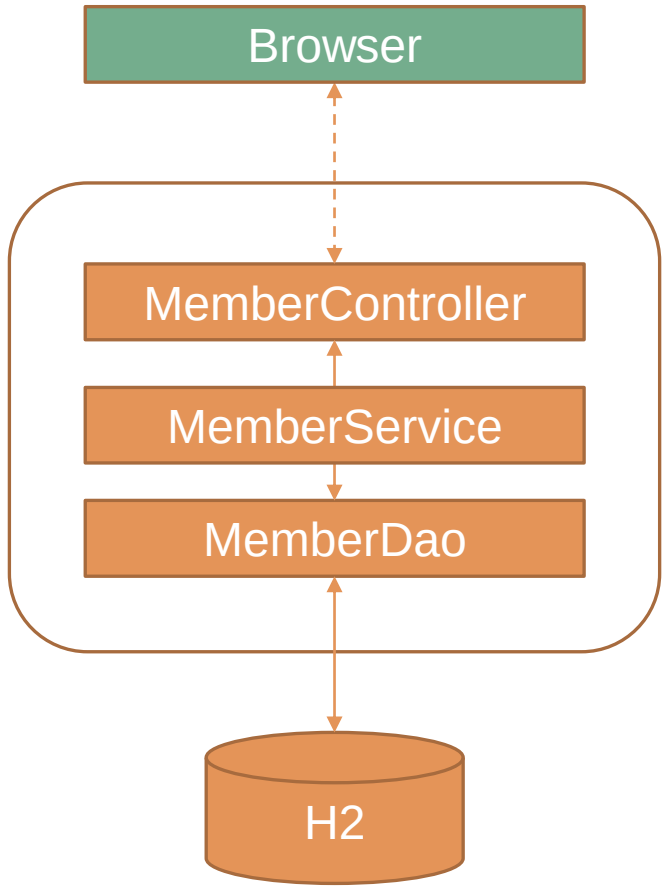


SpringBoot Additional Missions

이상현

Mission A

- 내용
  - Mission3에서 데이터베이스 접근 방식 변경
  - MemberDao에서 Connection 객체를 직접 생성하는 방법이 아니라 DataSource를 Autowired 한 뒤 Conneciton 객체를 얻는 방식
    - @Autowired
    - private DataSource dataSource;
    - Connection con = dataSource.getConnection();
  - src/main/resources/application.properties 설정 추가
    - `spring.datasource.driver-class-name=org.h2.Driver`
    - `spring.datasource.url=jdbc:h2:tcp://localhost/~/mission3`
    - `spring.datasource.username=sa`
    - `spring.datasource.password=`



Mission B

- 내용
  - Mission3에서 데이터베이스 접근 방식 변경
  - Dao에서 Connection 객체를 만들 필요없이 JdbcTemplate 객체를 Autowired로 얻어서 질의를 하는 방식
    - @Autowired
    - private JdbcTemplate jdbcTemplate;
    - jdbcTemplate.query(…);
  - src/main/resources/application.properties 설정 추가
    - `spring.datasource.driver-class-name=org.h2.Driver`
    - `spring.datasource.url=jdbc:h2:tcp://localhost/~/mission3`
    - `spring.datasource.username=sa`
    - `spring.datasource.password=`