회원 등록 및 리스트, 상세보기

1. 프로젝트 생성 : 스타터 사이트로 이동해서 스프링 프로젝트 생성 https://start.spring.io

Project : Gradle – Groovy

Language : Java

SpringBoot ; 2.7.11

Packaging : Jar

Java : 8

Dependencies :

spring web / Thymeleaf / Lombok / Spring Boot DevTools

1. 인텔리J에서 build.gradle로 open as Project
2. settings > Build,Execution,Deployment > BuildTools > Gradle > Build and run using과 Run Tests using을 IntelliJ로 바꾸면 속도가 조금 빠름
3. application.properties

server.port=8090

1. model(dto, vo)

package com.lec.ch04\_member.model;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
@Data  
@NoArgsConstructor  
public class Member {  
 private int no;  
 private String name;  
 private String tel;  
 public Member(String name, String tel) {  
 this.name = name;  
 this.tel = tel;  
 }  
}

1. Repository(Dao)

package com.lec.ch04\_member.repository;  
import com.lec.ch04\_member.model.Member;  
import java.util.List;  
import java.util.Optional;  
public interface **MemberRepository** {  
 public void save(Member member);  
 public Optional<Member> findByNo(int no);  
 public Optional<Member> findByName(String name);  
 public List<Member> findAll();  
}

package com.lec.ch04\_member.repository;  
import com.lec.ch04\_member.model.Member;  
import org.springframework.stereotype.Repository;  
import java.util.\*;  
@Repository  
public class MemoryMemberRepository implements MemberRepository {  
 private static Map<Integer, Member> *store* = new HashMap<>();  
 private static int *sequence* = 0;  
 @Override  
 public void save(Member member) {  
 member.setNo(++*sequence*);  
 *store*.put(member.getNo(), member);  
 }  
 @Override  
 public Optional<Member> findByNo(int no) {  
 return Optional.*ofNullable*(*store*.get(no));  
 }  
 @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();  
 }  
}

1. Service

import com.lec.ch04\_member.model.Member;  
import com.lec.ch04\_member.repository.MemberRepository;  
import com.lec.ch04\_member.repository.MemoryMemberRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import java.util.List;  
import java.util.Optional;  
@Service  
public class MemberService { *//ctrl+shift+t를 하면 create new test* private MemoryMemberRepository memberRepository;  
 @Autowired  
 public MemberService(MemoryMemberRepository memberRepository) {  
 this.memberRepository = memberRepository;  
 }  
  
 *// 회원가입* public int join(Member member){  
 *// 같은 이름이 있는 중복 회원 X  
// Optional<Member> result = memberRepository.findByName(member.getName());  
// result.ifPresent(m -> {  
// throw new IllegalStateException("이미 존재하는 회원입니다");  
// });  
 /\*memberRepository.findByName(member.getName()).ifPresent(member1 -> {  
 throw new IllegalStateException("이미 존재하는 회원입니다");  
 });\*/* validateDuplicateMember(member); *// 중복 회원 검증* memberRepository.save(member);  
 return member.getNo();  
 }  
 private void validateDuplicateMember(Member member){  
 memberRepository.findByName(member.getName()).ifPresent(member1 -> {  
 throw new IllegalStateException("이미 존재하는 회원입니다");  
 });  
 }  
 public List<Member> findMembers(){ *// 전체 회원 조회* return memberRepository.findAll();  
 }  
 public Optional<Member> findOne(int no){  
 return memberRepository.findByNo(no);  
 }  
 public void clearStore(){  
 memberRepository.clearStore();  
 }  
}

1. 단위 테스트

package com.lec.ch04\_member.repository;  
import com.lec.ch04\_member.model.Member;  
import org.junit.jupiter.api.AfterEach;  
import org.junit.jupiter.api.Test;  
import java.util.List;  
import static org.assertj.core.api.Assertions.\*;  
public class MemoryMemberRepositoryTest {  
 MemoryMemberRepository repository = new MemoryMemberRepository();  
 @AfterEach  
 public void afterEach(){  
 repository.clearStore();  
 }  
 @Test  
 public void save(){  
 Member member = new Member();  
 member.setName("홍길동");member.setTel("010-9999-9999");  
 repository.save(member);  
 Member result = repository.findByNo(member.getNo()).get();  
 System.*out*.println("save");  
 *//System.out.println("result : " + result.equals(member));  
 //Assertions.assertEquals(result, member);  
 assertThat*(member).isEqualTo(result);  
 }  
 @Test  
 public void findByName(){  
 Member member1 = new Member( "홍길자", "010-8888-8888");  
 repository.save(member1);  
 Member member2 = new Member( "홍길순", "010-8888-8888");  
 repository.save(member2);  
 Member result = repository.findByName("홍길순").get();  
 System.*out*.println("findByName");  
 *assertThat*(result).isEqualTo(member2);  
 }  
 @Test  
 public void findAll(){  
 Member member1 = new Member( "홍길자", "010-8888-8888");  
 repository.save(member1);  
 Member member2 = new Member( "홍길순", "010-8888-8888");  
 repository.save(member2);  
 List<Member> result = repository.findAll();  
 System.*out*.println("findAll");  
 *assertThat*(result.size()).isEqualTo(2);  
 }  
}

package com.lec.ch04\_member.service;  
import com.lec.ch04\_member.model.Member;  
import static org.junit.jupiter.api.Assertions.*assertThrows*;  
import com.lec.ch04\_member.repository.MemoryMemberRepository;  
import org.junit.jupiter.api.AfterEach;  
import org.junit.jupiter.api.Test;  
*//import static org.junit.jupiter.api.Assertions.\*;*import static org.assertj.core.api.Assertions.\*;  
class **MemberServiceTest** {  
 private MemberService memberService= new MemberService(new MemoryMemberRepository());  
  
 @AfterEach  
 public void afterEach(){  
 System.*out*.println("AfterEach");  
 memberService.clearStore();  
 }  
 @Test  
 void 회원가입() {  
 *//given* Member member = new Member("홍길자", "010-9999-9999");  
 *// when* int saveId = memberService.join(member);  
 *// then  
 assertThat*(saveId).isEqualTo(member.getNo());  
 Member findMember = memberService.findOne(saveId).get();  
 *assertThat*(member.getName()).isEqualTo(findMember.getName());  
 }  
 @Test  
 void 중복회원\_예외(){  
 *// given* Member member1 = new Member("홍길자", "010-9999-9999");  
 Member member2 = new Member("홍길자", "010-8888-9999");  
 *// when* memberService.join(member1);  
 */\*try {  
 memberService.join(member2);  
 fail("중복");  
 }catch (IllegalStateException e){  
 assertThat(e.getMessage()).isEqualTo("이미 존재하는 회원입니다");  
 }\*/* Exception e = *assertThrows*(IllegalStateException.class, ()->memberService.join(member2));  
 *// then  
 assertThat*(e.getMessage()).isEqualTo("이미 존재하는 회원입니다");  
 }  
}

1. 컨트롤러

package com.lec.ch04\_member.controller;  
import com.lec.ch04\_member.model.Member;  
import com.lec.ch04\_member.service.MemberService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PostMapping;  
import java.io.UnsupportedEncodingException;  
import java.net.URLEncoder;  
@Controller  
public class MemberController {  
 private MemberService memberService;  
 @Autowired  
 public MemberController(MemberService memberService) {  
 this.memberService = memberService;  
 }  
 *//@RequestMapping(value="/", method = RequestMethod.GET)* @GetMapping(value="/")  
 public String home(Model model){  
 model.addAttribute("greeting", "첫화면입니다. 반갑습니다");  
 return "home";  
 }  
 @GetMapping("/join")  
 public String joinView(){  
 return "member/join";  
 }  
 @PostMapping("/join")  
 public String join(Member member) throws UnsupportedEncodingException {  
 try {  
 memberService.join(member);  
 }catch (IllegalStateException e){  
 System.*out*.println(e.getMessage());  
 String msg = URLEncoder.*encode*("중복된 이름은 등록 불가합니다","utf-8");  
 return "redirect:join?msg="+msg;  
 }  
 return "redirect:/";  
 }  
 @GetMapping(value = "list")  
 public String list(Model model){  
 model.addAttribute("memberList", memberService.findMembers());  
 return "member/list";  
 }  
 @GetMapping(value="findOne")  
 public String findOne(int no, Model model){  
 model.addAttribute("member", memberService.findOne(no).get());  
 return "member/findOne";  
 }  
}

1. View(home.html)

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
 <link href="/css/style.css" rel="stylesheet" type="text/css">  
</head>  
<body>  
 <h1 th:text="${greeting}">Hello, Spring</h1>  
 <p>  
 <button onclick="location.href='/join'">회원가입</button>  
 <button onclick="location.href='/list'">회원목록</button>  
 </p>  
</body>  
</html>

1. View(member/join.html)

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
 <link href="/css/style.css" rel="stylesheet" type="text/css">  
</head>  
<body>  
 <script th:inline="javascript">  
 let msg = [[${param.msg}]];  
 if(msg){  
 alert(msg);  
 }  
 </script>  
 <form action="/join" method="post">  
 <table>  
 <tr><td><input type="text" name="name" placeholder="이름을 입력하세요"></td></tr>  
 <tr><td><input type="text" name="tel" placeholder="전화번호를 입력하세요"></td></tr>  
 <tr>  
 <td>  
 <button onclick="location.href='/join'">회원가입</button>  
 <button onclick="location.href='/list'">회원목록</button>  
 <button onclick="location.href='/'">처음으로</button>  
 </td>  
 </tr>  
 </table>  
 </form>  
</body>  
</html>

1. View(member/list.html)

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
 <link href="/css/style.css" rel="stylesheet" type="text/css">  
 <script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>  
 <script>  
 $(document).ready(function(){  
 $('tr').click(function(){  
 let no = Number($(this).children().eq(0).text());  
 if(!isNaN(no)){  
 location.href = '/findOne?no='+no;  
 }  
 });  
 });  
 </script>  
</head>  
<body>  
 <table>  
 <tr>  
 <td colspan="3">  
 <a href="/join">회원가입</a> &nbsp; &nbsp; &nbsp; &nbsp; &nbsp;  
 <a href="/">처음으로</a>  
 </td>  
 </tr>  
 <tr>  
 <th>번호</th><th>이름</th><th>전화번호</th>  
 </tr>  
 <tr th:if="${memberList.size() == 0}">  
 <td colspan="3">  
 <b>등록된 회원이 없습니다</b>  
 </td>  
 </tr>  
 <tr th:each="member:${memberList}">  
 <td th:text="${member.no}">#no</td>  
 <td th:text="${member.name}">#name</td>  
 <td th:text="${member.tel}">#tel</td>  
 </tr>  
 <tr th:unless="${memberList.size() == 0}">  
 <td colspan="3" th:text="'이상 &nbsp; ' + ${memberList.size()} + '명'">xx명</td>  
 </tr>  
 </table>  
</body>  
</html>

1. View(member/findOne.html)

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
 <link href="/css/style.css" rel="stylesheet" type="text/css">  
</head>  
<body>  
 <table>  
 <tr><td>번호</td><td th:text="${member.no}">#no</td></tr>  
 <tr><td>이름</td><td th:text="${member.name}">#name</td></tr>  
 <tr><td>전화번호</td><td th:text="${member.tel}">#tel</td></tr>  
 <tr>  
 <td colspan="2">  
 <input type="submit" value="등록" class="btn">  
 <input type="button" value="목록" onclick="location.href='/list'" class="btn">  
 <input type="button" value="처음으로" onclick="location.href='/'" class="btn">  
 </td>  
 </tr>  
 </table>  
</body>  
</html>