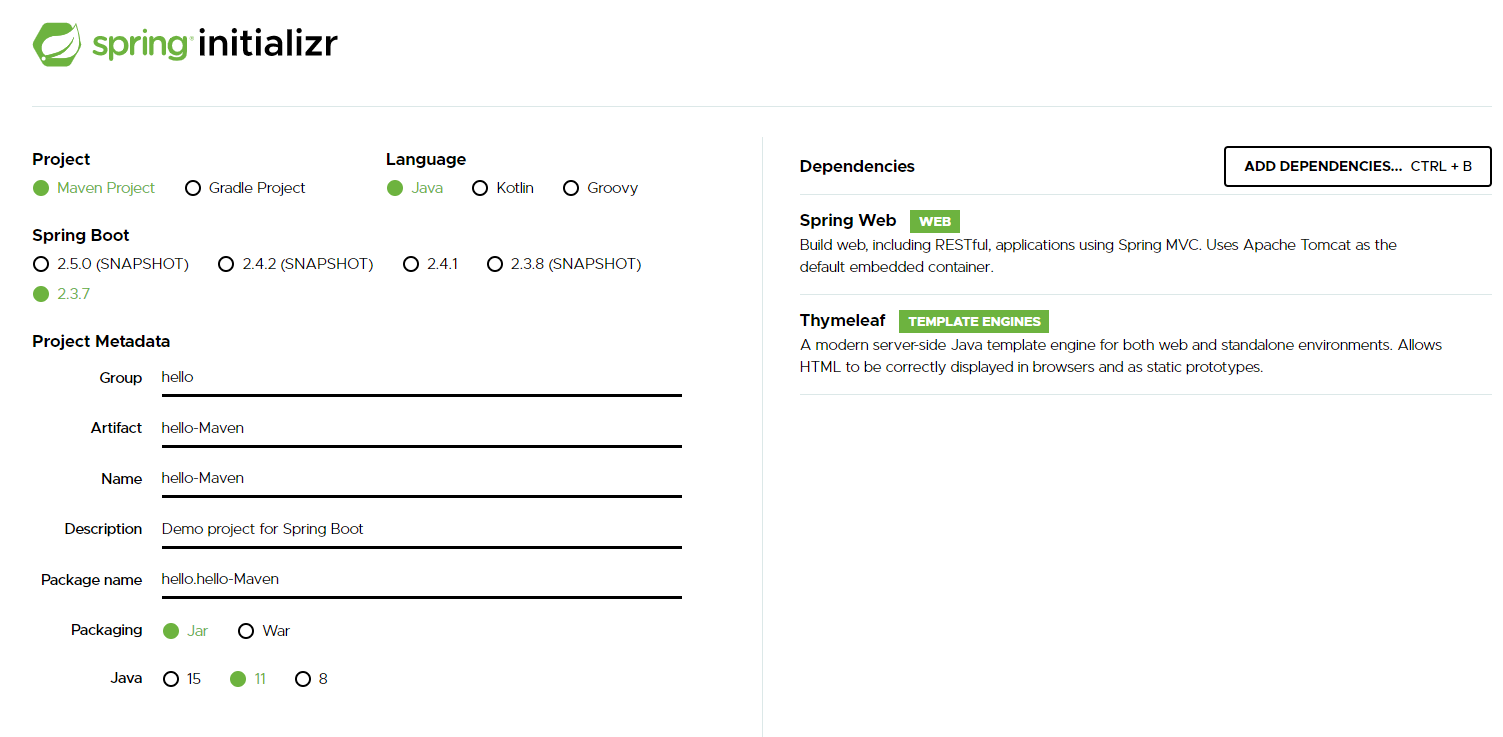
//3-2

Spring Maven

https://start.spring.io/

프로젝트 생성 후 intellij에서 File -> opne -> 프로젝트 -> ok

1. 기본적인 웹 동작

Hello.helloMaven.controller(생성) ->HomeController.java 생성 -> 내용 입력



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

Src/main/resources/templates -> home.html 생성 -> 내용 입력



<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org">  
<body>  
<div class="container">  
 <div>  
 <h1>Hello Spring</h1>  
 <p>회원 기능</p>  
 <p>  
 <a href="/members/new">회원 가입</a>  
 <a href="/members">회원 목록</a>  
 </p>  
 </div>  
</div> <!-- /container -->  
</body>  
</html>

2. 도메인, 저장소 생성

Hello.helloMaven.domain(생성) -> Member.java(생성)



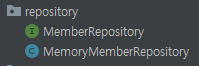
public class Member {  
  
 private Long id;  
  
 private String name;  
  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  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

Hello.helloMaven.repository(생성) -> MemberRepository.java(인터페이스)(생성)



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

Hello.helloMaven.repository -> MemoryMemberRepository.java(생성) //메모리 구현체



public class MemoryMemberRepository implements MemberRepository{  
  
 private static Map<Long, Member> *store* = new HashMap<>();  
 private static long *sequence* = 8L;  
  
 @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();  
 }  
}

3. 서비스 개발

Hello.helloMaven.service(생성) -> MemberService.java



public class MemberService {  
  
 private final MemberRepository memberRepository;  
  
 //@Autowired  
 public MemberService(MemberRepository memberRepository) {  
 this.memberRepository = memberRepository;  
 }  
  
 //회원가입  
 public long join(Member member) {  
 //같은 이름이 있는 중복 회원x  
 validateDuplicateMember(member);//중복회원검증  
 memberRepository.save(member);  
 return member.getId();  
 }  
  
 private void validateDuplicateMember(Member member) {  
 memberRepository.findByName(member.getName())  
 .ifPresent(m -> {  
 throw new IllegalStateException("이미 존재하는 회원입니다.");  
 });  
 }  
 //전체 회원 조회  
 public List<Member> findMembers(){  
 return memberRepository.findAll();  
 }  
  
 public Optional<Member> findOne(Long memberId){  
 return memberRepository.findById((memberId));  
 }  
}

4. 등록기능

Hello.helloMaven.controller -> MemberController.java(생성)



public class MemberController {  
 private final MemberService memberService;  
  
 public MemberController(MemberService memberService) {  
 this.memberService = memberService;  
 }  
  
 @GetMapping("/members/new")  
 public String createForm() {  
 return "members/createMemberForm";  
 }  
}

등록기능 html 추가

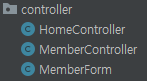
Templates.members(생성) -> createMemberForm.html(생성) -> 내용입력



<!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> <!-- /container -->  
</body>  
</html>

등록화면에서 데이터를 전달 받을 객체

Hello.helloMaven.controller -> MemberForm.java(생성) -> 내용입력



public class MemberForm {  
  
 private String name;  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
}

실제로 등록하는 기능 // Hello.helloMaven.controller -> MemberController.java

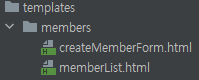
@PostMapping("/members/new")  
public String create(MemberForm form) {  
  
 Member member = new Member();  
 member.setName(form.getName());  
  
 memberService.join(member);  
  
 return "redirect:/";  
}

5. 회원 리스트 출력

Hello.helloMaven.controller -> MemberController.java

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

Templates.members -> memberList.html



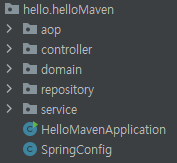
<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org">  
<body>  
<div class="container">  
 <div>  
 <table>  
 <thead>  
 <tr>  
 <th>#</th>  
 <th>이름</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr th:each="member : ${members}">  
 <td th:text="${member.id}"></td>  
 <td th:text="${member.name}"></td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
</div> <!-- /container -->  
</body>  
</html>

6. 스프링 빈과 의존관계 설정

Hello.helloMaven.controller -> MemberController.java @추가

@Controller  
public class MemberController {  
 private final MemberService memberService;  
  
 @Autowired  
 public MemberController(MemberService memberService) {  
 this.memberService = memberService;  
 }

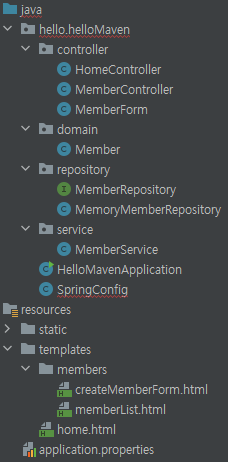
Hello.helloMaven -> SPringConfig



@Configuration  
public class SpringConfig {

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

구조



7. jdbc. Mysql을 사용한 회원등록, 회원리스트출력

Jdbc와 mysql을 사용하기 위해서 pom.xml에 내용을 추가한다 //<dependencies>내부에 추가

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-jdbc</artifactId>  
</dependency>

<dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
</dependency>

Application.properties -> 내용추가 //공백에 주의

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
spring.datasource.url=jdbc:mysql://주소명:port번호/database명?useSSL=false&characterEncoding=UTF-8&serverTimezone=UTC  
spring.datasource.username=ID  
spring.datasource.password=PW  
  
spring.jdbc.show-sql=true  
spring.jdbc.generate-ddl=true  
spring.jdbc.database=mysql

hello.helloMaven.repository -> JdbcMemberRepository.java(생성)



public class JdbcMemberRepository implements MemberRepository{  
  
 private final DataSource dataSource;  
 public JdbcMemberRepository(DataSource dataSource) {  
 this.dataSource = dataSource;  
 }  
 @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,  
 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 조회 실패");  
 }  
 return member;  
 } catch (Exception e) {  
 throw new IllegalStateException(e);  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 }  
 @Override  
 public Optional<Member> findById(Long id) {  
 String sql = "select \* from member where id = ?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setLong(1, id);  
 rs = pstmt.executeQuery();  
 if(rs.next()) {  
 Member member = new Member();  
 member.setId(rs.getLong("id"));  
 member.setName(rs.getString("name"));  
 return Optional.*of*(member);  
 } else {  
 return Optional.*empty*();  
 }  
 } catch (Exception e) {  
 throw new IllegalStateException(e);  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 }  
 @Override  
 public List<Member> findAll() {  
 String sql = "select \* from member";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 rs = pstmt.executeQuery();  
 List<Member> members = new ArrayList<>();  
 while(rs.next()) {  
 Member member = new Member();  
 member.setId(rs.getLong("id"));  
 member.setName(rs.getString("name"));  
 members.add(member);  
 }  
 return members;  
 } catch (Exception e) {  
 throw new IllegalStateException(e);  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 }  
 @Override  
 public Optional<Member> findByName(String name) {  
 String sql = "select \* from member where name = ?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setString(1, name);  
 rs = pstmt.executeQuery();  
 if(rs.next()) {  
 Member member = new Member();  
 member.setId(rs.getLong("id"));  
 member.setName(rs.getString("name"));  
 return Optional.*of*(member);  
 }  
 return Optional.*empty*();  
 } catch (Exception e) {  
 throw new IllegalStateException(e);  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 }  
 private Connection getConnection() {  
 return DataSourceUtils.*getConnection*(dataSource);  
 }  
 private void close(Connection conn, PreparedStatement pstmt, ResultSet rs)  
 {  
 try {  
 if (rs != null) {  
 rs.close();  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 try {  
 if (pstmt != null) {  
 pstmt.close();  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 try {  
 if (conn != null) {  
 close(conn);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 private void close(Connection conn) throws SQLException {  
 DataSourceUtils.*releaseConnection*(conn, dataSource);  
 }  
}

hello.helloMaven -> SpringConfig

@Configuration  
public class SpringConfig {  
  
 private final DataSource dataSource;  
  
  
 public SpringConfig(DataSource dataSource) {  
 this.dataSource = dataSource;  
  
 }  
  
 @Bean  
 public MemberService memberService() {  
 return new MemberService(memberRepository());  
 }  
  
 @Bean  
 public MemberRepository memberRepository() {  
  
 return new JdbcMemberRepository(dataSource);

}  
}

8. JdbcTemplate // 라이브러리 추가 x

Hello.helloMaven.repository -> JdbcTemplateMemberRepository.java



public class JdbcTemplateMemberRepository implements MemberRepository {  
  
 private final JdbcTemplate jdbcTemplate;  
  
 public JdbcTemplateMemberRepository(DataSource dataSource) {  
 this.jdbcTemplate = new JdbcTemplate(dataSource);  
 }  
  
 @Override  
 public Member save(Member member) {  
 SimpleJdbcInsert jdbcInsert = new SimpleJdbcInsert(jdbcTemplate);  
 jdbcInsert.withTableName("member").usingGeneratedKeyColumns("id");//테이블 사용 키  
 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();  
 }  
  
 @Override  
 public List<Member> findAll() {  
 return jdbcTemplate.query("select \* from member", memberRowMapper());  
 }  
  
 private RowMapper<Member> memberRowMapper() {  
 return (rs, rowNum) -> {  
 Member member = new Member();  
 member.setId(rs.getLong("id"));  
 member.setName(rs.getString("name"));  
 return member;  
 };  
 }  
}

hello.helloMaven -> SpringConfig

@Configuration  
public class SpringConfig {  
  
 private final DataSource dataSource;  
  
 public SpringConfig(DataSource dataSource) {  
 this.dataSource = dataSource;  
 this.entityManager = entityManager;  
 }

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

9. JPA

Pom.xml -> 추가

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
</dependency>

Hello.helloMaven.domain -> Member @추가 //엔티티매핑

@Entity //Jpa 추가  
public class Member {  
  
 @Id @GeneratedValue(strategy = GenerationType.*IDENTITY*) //Jpa 추가  
 private Long id;  
  
 private String name;

Hello.helloMaver.repository -> JpaMemberRepository(생성)

public class JpaMemberRepository implements MemberRepository {  
  
 private final EntityManager entityManager;  
  
 public JpaMemberRepository(EntityManager entityManager) {  
 this.entityManager = entityManager;  
 }  
  
 @Override  
 public Member save(Member member) {  
 entityManager.persist(member);  
 return member;  
 }  
  
 @Override  
 public Optional<Member> findById(Long id) {  
 Member member = entityManager.find(Member.class, id);  
 return Optional.*ofNullable*(member);  
 }  
  
 @Override  
 public Optional<Member> findByName(String name) {  
 List<Member> result = entityManager.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 entityManager.createQuery("select m from Member m", Member.class).getResultList();  
 }  
}

hello.helloMaven.SpringConfig -> 변경

@Configuration  
public class SpringConfig {  
  
 private final DataSource dataSource;  
 private final EntityManager entityManager;  
  
 public SpringConfig(DataSource dataSource, EntityManager entityManager) {  
 this.dataSource = dataSource;  
 this.entityManager = entityManager;  
 }  
  
 @Bean  
 public MemberService memberService() {  
 return new MemberService(memberRepository());  
 }  
  
 @Bean  
 public MemberRepository memberRepository() {  
 return new JpaMemberRepository(entityManager);  
 }  
}

10. Aop(Aspect Oriented Programming) 라이브러리추가 x

모든 메소드의 호출 시간을 측정하고 싶을 때 사용

Aop패키지 생성 -> TimeTraceAop.java 생성

@Aspect  
@Component  
public class TimeTraceAop {  
  
 @Around("execution (\* hello.helloMaven..\*(..))")//적용위치  
 public Object excute(ProceedingJoinPoint joinPoint) throws Throwable {  
 long start = System.*currentTimeMillis*();  
 System.*out*.println("START : " + joinPoint.toLongString());  
 try{  
 return joinPoint.proceed();  
 }finally {  
 long finish = System.*currentTimeMillis*();  
 long timeMs = finish - start;  
 System.*out*.println("END : " + joinPoint.toLongString() + " " + timeMs + "ms");  
 }  
  
 }  
}

실행내용

