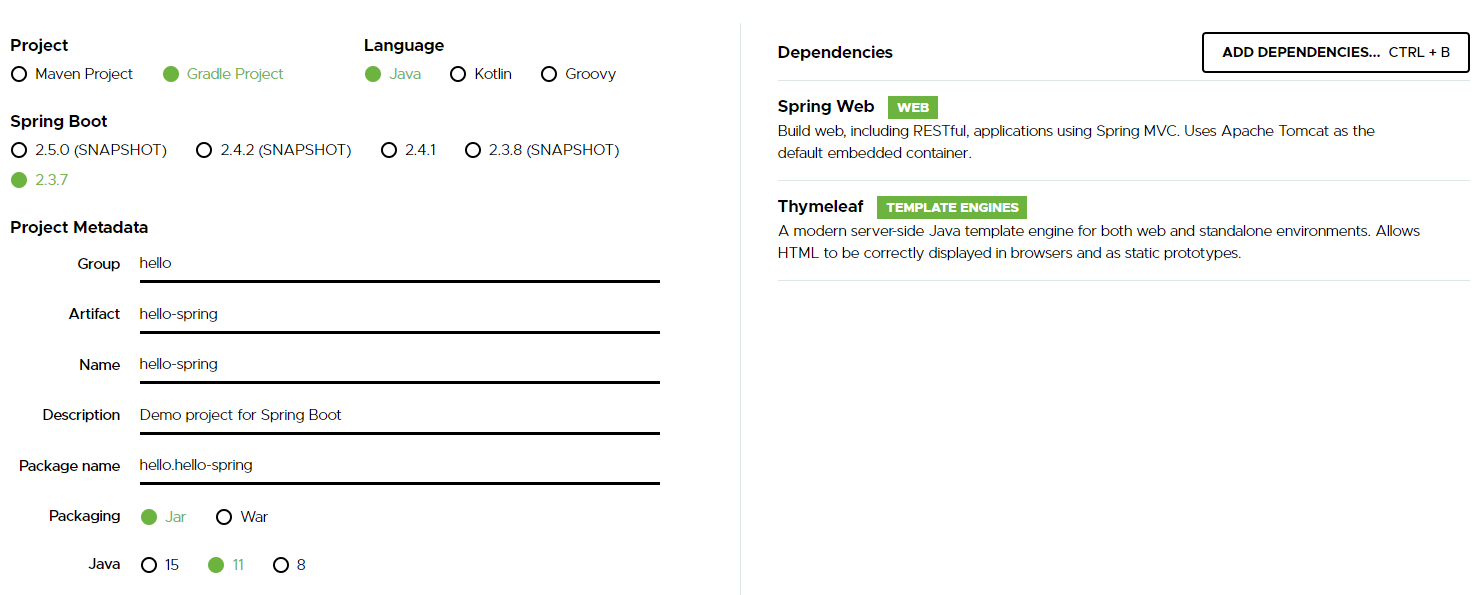
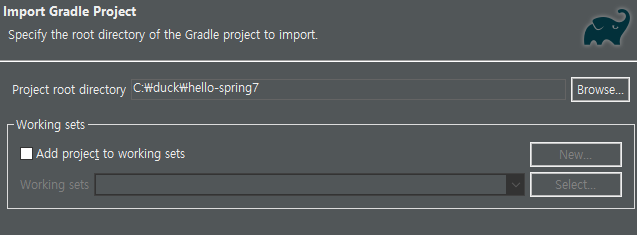
Eclipse에서 실습하기

준비물 java 11, Eclipse

예제와 동일하게 start.spring.io에서 프로젝트 생성



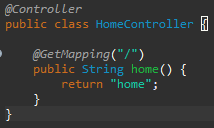
Eclipse -> import -> Gradle -> Existing Gradle Project -> 생성한 프로젝트 위치 -> Finish



1. 기본적인 웹 동작

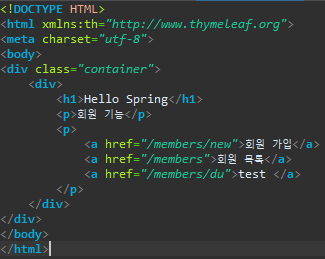
Hello.hellospring.controller(생성) ->HomeController.java 생성 -> 내용 입력



~~~~

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

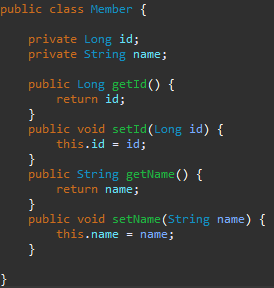




2. 도메인, 저장소 생성

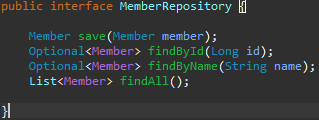
Hello.hellospring.domain(생성) -> Member.java(생성)



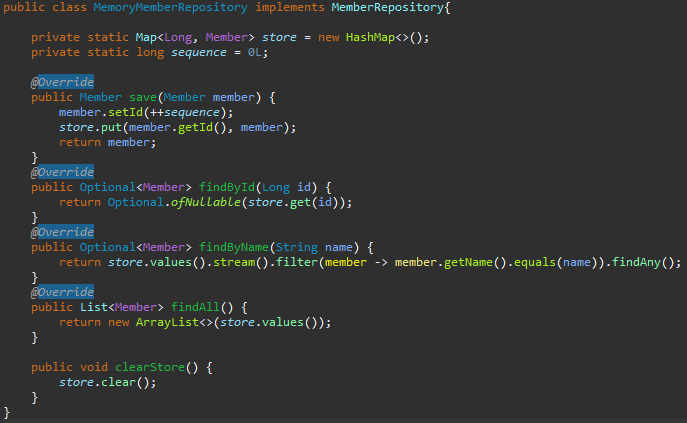


Hello.hellospring.repository(생성) -> MemberRepository.java(인터페이스)(생성)





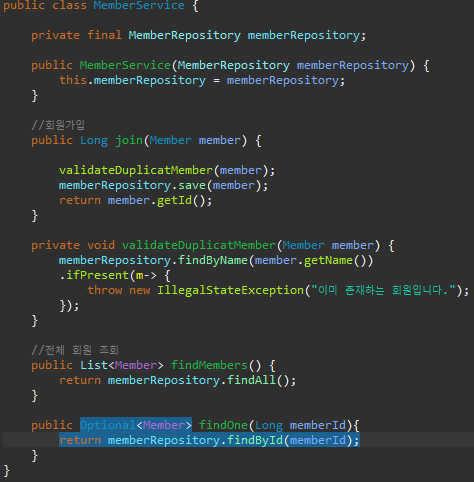
Hello.hellospring.repository -> MemoryMemberRepository.java(생성) //메모리 구현체



3. 서비스 개발

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

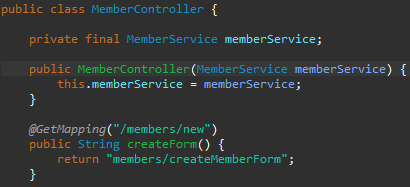




4. 등록기능

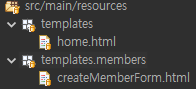
Hello.hellosrping.controller -> MemberController.java(생성)

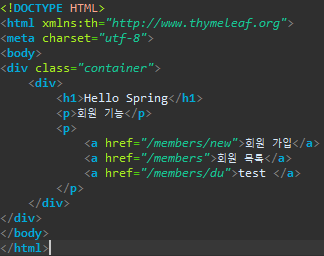




등록기능 html 추가

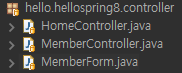
Templates.members(생성) -> createMemberForm.html(생성) -> 내용입력

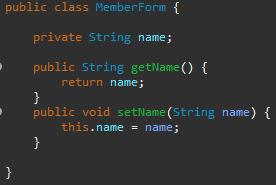




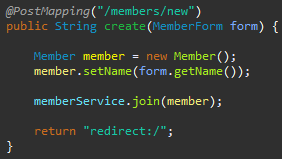
등록화면에서 데이터를 전달 받을 객체

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



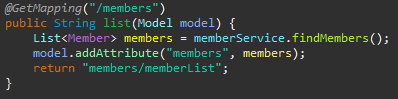


실제로 등록하는 기능 // Hello.hellosrping.controller -> MemberController.java



5. 회원 리스트 출력

Hello.hellosrping.controller -> MemberController.java



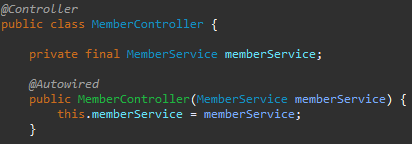
Templates.members -> memberList.html





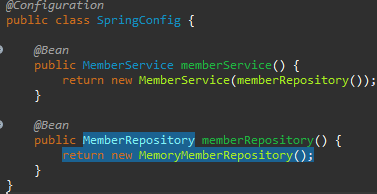
6. 스프링 빈과 의존관계 설정

Hello.hellosrping.controller -> MemberController.java

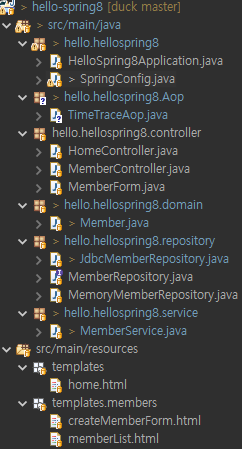


Hello.hellospring -> SPringConfig





구조



7. jdbc. Mysql을 사용한 회원등록, 회원리스트출력

Jdbc, mysql connector 설치 및 적용방법

(작성일 2021.01.08

Spring-boot-starter-jdbc-2.3.7.RELEASE.jar

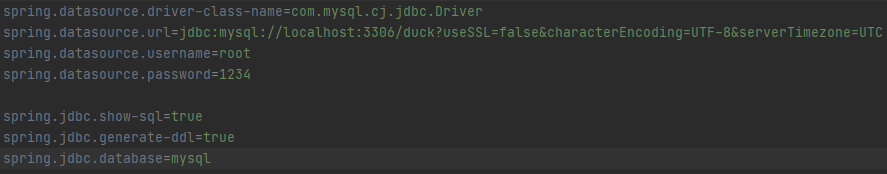
Spring-jdbc-5.2.12.RELEASE.jar

Mysql-connector-java-5.1.49.jar, Mysql-connector-java-5.1.49-bin.jar)

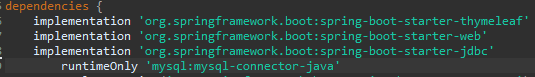
Jdbc와 mysql connector을 인터넷에 검색해서 다운 받은 후

프로젝트 -> 우클릭 -> Properties -> java build Path -> Add External JARs -> 다운받은 lib

-> Apply and Close -> 프로젝트 -> 우클릭 -> Gradle -> Refresh

db설정 application.properties//url-주소:port번호/database명 : 

build.gradle -> 설정

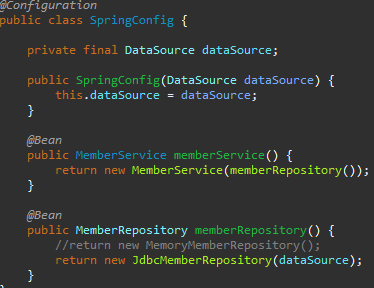


Hello.hellospring.repository -> JdbcMemberRepository.java –입력



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.hellosrping -> SpringConfig

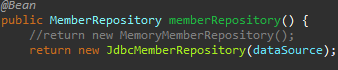


8. JdacTemplate

Hello.hellospring.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.hellosrping -> SpringConfig

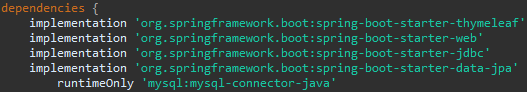


9. JPA

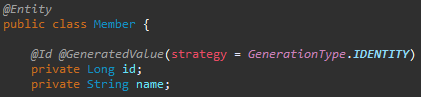
설정 (작성일 2021.01.08 spring-boot-starter-data-jpa-2.3.6.RELEASE.jar 사용)

Jdbc와 동일하게 라이브러리에 추가

Build.grradle -> jpa추가



Hello.helloisrping.domain -> Member,java @추가 //엔티티매핑



Hello.hellosrping.repository -> JpaMemberRepository(생성)

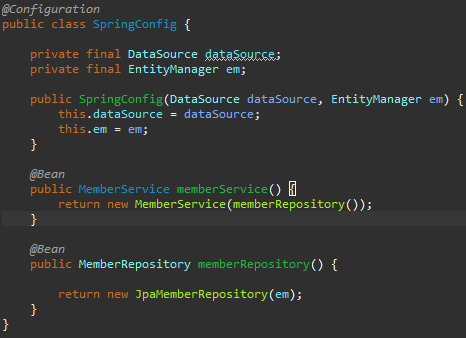


public class JpaMemberRepository implements MemberRepository {  
  
 private final EntityManager entityManager;  
  
 public JpaMemberRepository(EntityManager entityManager) {  
 this.entityManager = entityManager;  
 }  
  
 @Override  
 public Member save(Member member) {  
 entityManager.persist(member);  
 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.hellosrping.service -> MemberService @Transactional 추가



Hello.hellospring -> SpringConfig 추가



10. Aop(Aspect Oriented Programming)

모든 메소드의 호출 시간을 측정하고 싶을 때 사용

다른 라이브러리추가 x

Aop패키지 생성 -> TimeTraceAop.java 생성

