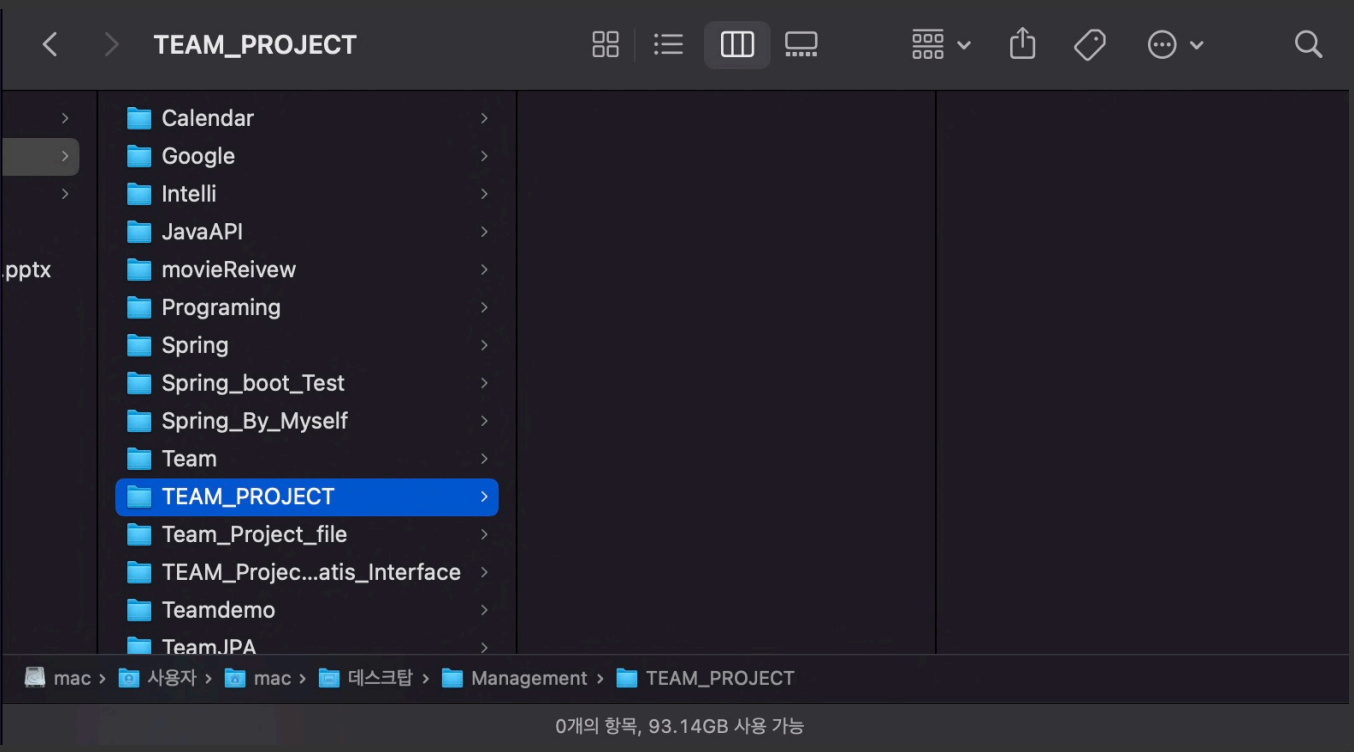
TEAM_
PROJECT_
BY_
MYSELF

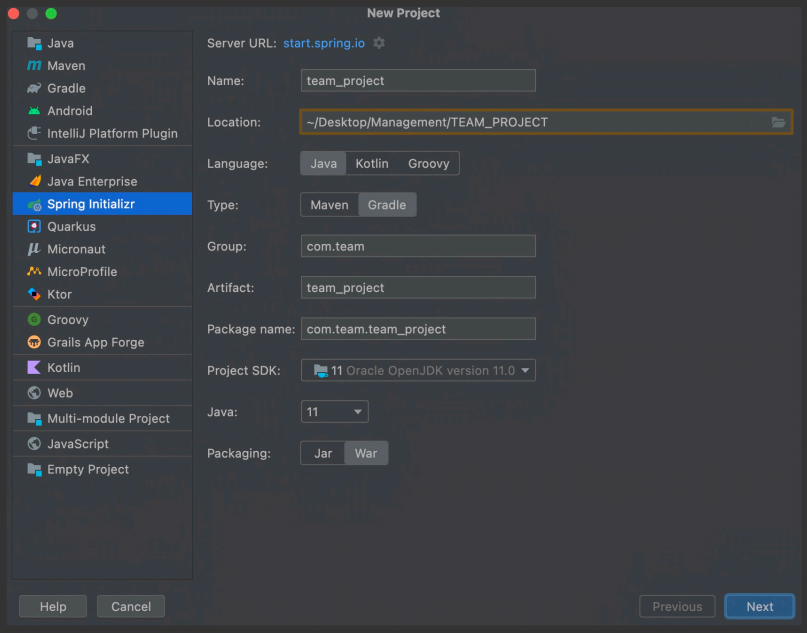
해당 Directory 사용



Project 생성

〈option〉

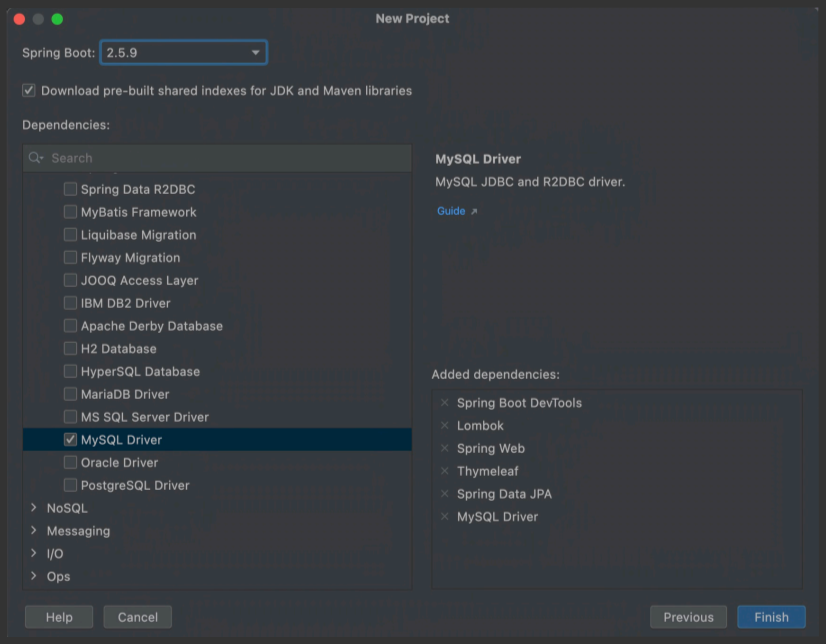
- name : team_project
- Language : Java(version : 11)
- Type : Gradle
- Group : com.team
- Packaging: war



Boot versiong : 2. 5. 9

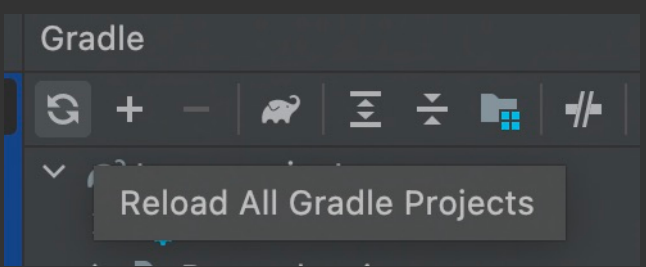
〈Dependencies〉

- Spring boot Devtools
- Lombok
- Spring Web
- Thymeleaf
- Spring Data JPA
- MySQL Driver



build.gradle

```
plugins {
    id 'org.springframework.boot' version '2.5.9'
    id 'io.spring.dependency-management' version '1.0.11.RELEASE'
    id 'java'
    id 'war'

    // Querydsl 을 위한 plug in
    id 'com.ewerk.gradle.plugins.querydsl' version '1.0.10'
}

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

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'mysql:mysql-connector-java'
    annotationProcessor 'org.projectlombok:lombok'
    providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'

    // Thymeleaf 를 위한 추가 의존성
    implementation group: 'org.thymeleaf.extras', name: 'thymeleaf-extras-java8time'


    // Querydsl
    implementation 'com.querydsl:querydsl-jpa'
    implementation 'com.querydsl:querydsl-apt'

}

test {
    useJUnitPlatform()
}


// querydsl 추가 시작
def querydslDir = "$buildDir/generated/querydsl"

querydsl {
    jpa = true
    querydslSourcesDir = querydslDir
}

sourceSets {
    main.java.srcDir querydslDir
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
    querydsl.extendsFrom compileClasspath
}

compileQuerydsl {
    options.annotationProcessorPath = configurations.querydsl
}
```

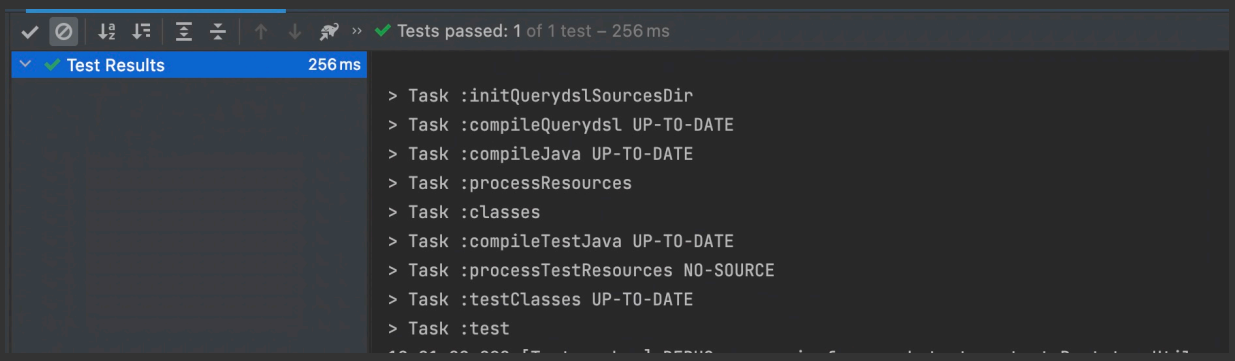


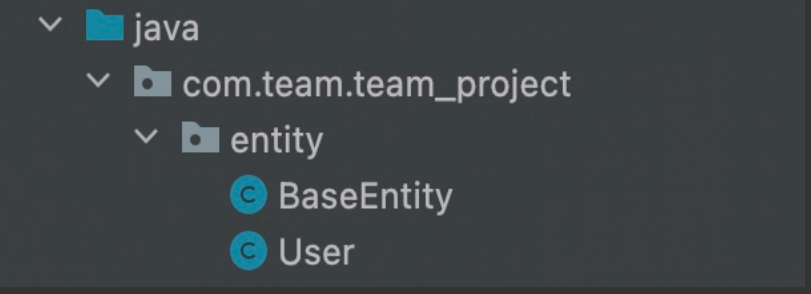
——〉
Reload All Gradle Projects 실행

application.properties

```
server.port=1996

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/TEAM
spring.datasource.username=singsiuk
spring.datasource.password=ssw0304

spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.show-sql=true

spring.thymeleaf.cache=false
```

result





Entity 생성

```java
package com.team.team_project.entity;

import lombok.*;
import org.hibernate.annotations.DynamicInsert;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import javax.persistence.*;
import java.time.LocalDate;

@Entity
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Getter
@ToString
@EntityListeners(value={AuditingEntityListener.class})
@Table(name="User",uniqueConstraints = {
        @UniqueConstraint(columnNames = {"email","id","nick"})})
@DynamicInsert
public class User{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long code;

    @Column(name="id", length = 255, nullable = false)
    private String id;

    @Column(name="email",length = 255, nullable = false)
    private String email;

    @Column(name="pw",length = 255, nullable = false)
    private String pw;

    @Column(name="nick",length = 50, nullable = false)
    private String nick;

    @Column(name="birthday",nullable = false)
    private LocalDate birthday;

    @Column(name="gender",columnDefinition = "varchar(2) check (status in ('m','f')")
    private String gender;


    @Column(columnDefinition = " varchar(5) default '회원'")
    private int status;

}
```

ERORR

```
org.hibernate.tool.schema.spi.CommandAcceptanceException: Error executing DDL "
    create table user (
       code bigint not null auto_increment,
        birthday date not null,
        email varchar(255) not null,
        gender varchar(2) check (status in ('m','f')), id varchar(255) not null, nick varchar(50) not null, pw varchar(255) not null, status varchar(5) default '회원', primary key (code)) engine=innoDB" via JDBC Statement
        at org.hibernate.tool.schema.internal.exec.GenerationTargetToDatabase.accept(GenerationTargetToDatabase.java:67) ~[hibernate-core-5.4.33.jar:5.4.33]
        at org.hibernate.tool.schema.internal.AbstractSchemaMigrator.applySqlString(AbstractSchemaMigrator.java:562) ~[hibernate-core-5.4.33.jar:5.4.33]
        at org.hibernate.tool.schema.internal.AbstractSchemaMigrator.applySqlStrings(AbstractSchemaMigrator.java:507) ~[hibernate-core-5.4.33.jar:5.4.33]
        at org.hibernate.tool.schema.internal.AbstractSchemaMigrator.createTable(AbstractSchemaMigrator.java:277) ~[hibernate-core-5.4.33.jar:5.4.33]
        at org.hibernate.tool.schema.internal.GroupedSchemaMigratorImpl.performTablesMigration(GroupedSchemaMigratorImpl.java:71) ~[hibernate-core-5.4.33.jar:5.4.33]
        at org.hibernate.tool.schema.internal.AbstractSchemaMigrator.performMigration(AbstractSchemaMigrator.java:207) ~[hibernate-core-5.4.33.jar:5.4.33]
        at org.hibernate.tool.schema.internal.AbstractSchemaMigrator.doMigration(AbstractSchemaMigrator.java:114) ~[hibernate-core-5.4.33.jar:5.4.33]
        at org.hibernate.tool.schema.spi.SchemaManagementToolCoordinator.performDatabaseAction(SchemaManagementToolCoordinator.java:184) ~[hibernate-core-5.4.33.jar:5.4.33]
        at org.hibernate.tool.schema.spi.SchemaManagementToolCoordinator.process(SchemaManagementToolCoordinator.java:73) ~[hibernate-core-5.4.33.jar:5.4.33]
        at org.hibernate.internal.SessionFactoryImpl.(init)(SessionFactoryImpl.java:318) ~[hibernate-core-5.4.33.jar:5.4.33]
        at org.hibernate.boot.internal.SessionFactoryBuilderImpl.build(SessionFactoryBuilderImpl.java:468) ~[hibernate-core-5.4.33.jar:5.4.33]
        at org.hibernate.jpa.boot.internal.EntityManagerFactoryBuilderImpl.build(EntityManagerFactoryBuilderImpl.java:1259) ~[hibernate-core-5.4.33.jar:5.4.33]
```

User.java

```java
package com.team.team_project.entity;

import lombok.*;
import org.hibernate.annotations.DynamicInsert;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import javax.persistence.*;
import java.time.LocalDate;

@Entity
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Getter
@ToString
@EntityListeners(value={AuditingEntityListener.class})
@Table(name="User",uniqueConstraints = {
        @UniqueConstraint(columnNames = {"email","id","nick"})})
@DynamicInsert
public class User{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long code;

    @Column(name="id", length = 255, nullable = false)
    private String id;

    @Column(name="email",length = 255, nullable = false)
    private String email;

    @Column(name="pw",length = 255, nullable = false)
    private String pw;

    @Column(name="nick",length = 50, nullable = false)
    private String nick;

    @Column(name="birthday",nullable = false)
    private LocalDate birthday;

//    @Column(name="gender",columnDefinition = "varchar(2) check (status in ('m','f')")
//    private String gender;

    @Column(columnDefinition = " varchar(5) default '회원'")
    private int status;

}
```

원인

```
gender varchar(2) check(gender in('M','F')) not null,
```

```
@Column(name="gender",columnDefinition = "varchar(2) check (gender in ('m','f')" ,nullable = false)
private String gender;
```

```
    alter table user
       add column gender varchar(2) check (gender in ('m','f') not null
2022-01-21 19:06:16.114  WARN 50424 --- [  restartedMain] o.h.t.s.i.ExceptionHandlerLoggedImpl
    alter table user
       add column gender varchar(2) check (gender in ('m','f') not null" via JDBC Statement

org.hibernate.tool.schema.spi.CommandAcceptanceException: Error executing DDL "
    alter table user
       add column gender varchar(2) check (gender in ('m','f') not null" via JDBC Statement <12 i
    at org.springframework.orm.jpa.vendor.SpringHibernateJpaPersistenceProvider.createContainerEn
```

```
@Column(name="gender",columnDefinition = "varchar(2) check (gender in ('m','f'))" ,nullable = false)
private String gender;
```

Success

```
2022-01-21 19:14:03.376  INFO 51109 --- [  restartedMain] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.56]
2022-01-21 19:14:03.449  INFO 51109 --- [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-01-21 19:14:03.449  INFO 51109 --- [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1463 ms
2022-01-21 19:14:03.677  INFO 51109 --- [  restartedMain] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: default]
2022-01-21 19:14:03.703  INFO 51109 --- [  restartedMain] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.4.33
2022-01-21 19:14:03.770  INFO 51109 --- [  restartedMain] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2022-01-21 19:14:03.816  INFO 51109 --- [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2022-01-21 19:14:04.198  INFO 51109 --- [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2022-01-21 19:14:04.210  INFO 51109 --- [  restartedMain] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
Hibernate:

    alter table user
       add column gender varchar(2) check (gender in ('m','f')) not null
2022-01-21 19:14:04.810  INFO 51109 --- [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine
2022-01-21 19:14:04.818  INFO 51109 --- [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default
2022-01-21 19:14:04.885  WARN 51109 --- [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database
2022-01-21 19:14:05.441  INFO 51109 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2022-01-21 19:14:05.486  INFO 51109 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 1996 (http) with context path ''
2022-01-21 19:14:05.505  INFO 51109 --- [  restartedMain] c.t.team_project.TeamProjectApplication  : Started TeamProjectApplication in 4.063 seconds (JVM running for 5
```

MySql_properties

```
CREATE TABLE `user` (
  `code` bigint NOT NULL AUTO_INCREMENT,
  `birthday` date NOT NULL,
  `email` varchar(255) NOT NULL,
  `id` varchar(255) NOT NULL,
  `nick` varchar(50) NOT NULL,
  `pw` varchar(255) NOT NULL,
  `status` varchar(5) DEFAULT '회원',
  `gender` varchar(2) NOT NULL,
  PRIMARY KEY (`code`),
  UNIQUE KEY `UK8p307bs5tiutxsu8988ift85` (`email`,`id`,`nick`),
  CONSTRAINT `user_chk_1` CHECK ((`gender` in (_utf8mb4'm',_utf8mb4'f')))
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

JPA 에서 Check Constraint 주는 방법

```
@Column(name="gender",
        columnDefinition = "varchar(2) check (gender in ('m','f'))" ,
        nullable = false)
private String gender;
```

나중에 수정 가능한 항목들을 Setter 지정 — User.java

```
public void changePw(String user_pw){
    this.pw = user_pw;
}

public void changeNick(String nick){
    this.nick = nick;
}

public void changeGender(String gender){
    this.gender = gender;
}

public void changeStatus(int status) {
    this.status = status;
}

public void changeBirthday(LocalDate birthday) {
    this.birthday = birthday;
}
```

register , update 시간 기록을 하기위해서 BaseEntity 를 생성

BaseEntity.java

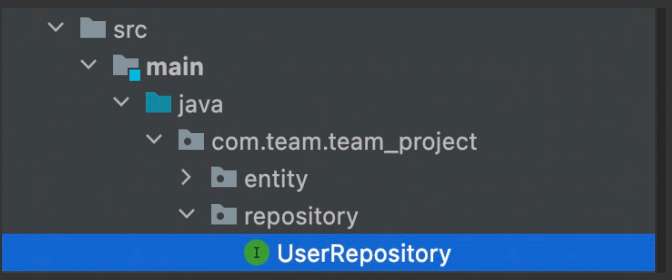
```java
package com.team.team_project.entity;

import lombok.Getter;
import org.springframework.data.annotation.CreatedDate;
import org.springframework.data.annotation.LastModifiedDate;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import javax.persistence.Column;
import javax.persistence.EntityListeners;
import javax.persistence.MappedSuperclass;
import java.time.LocalDateTime;

@MappedSuperclass
@EntityListeners(value = {AuditingEntityListener.class})
@Getter
public class BaseEntity {
    @CreatedDate
    @Column(name="regdate", updatable = false)
    private LocalDateTime regDate;

    @LastModifiedDate
    @Column(name="moddate")     //자동 업데이트
    private LocalDateTime modDate;
}
```

User. java 에서 상속 받기

User.java

```java
package com.team.team_project.entity;

import lombok.*;
import org.hibernate.annotations.DynamicInsert;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import javax.persistence.*;
import java.time.LocalDate;

@Entity
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Getter
@ToString
@EntityListeners(value={AuditingEntityListener.class})
@Table(name="User",uniqueConstraints = {
        @UniqueConstraint(columnNames = {"email","id","nick"})})
@DynamicInsert
public class User extends BaseEntity{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long code;

    @Column(name="id", length = 255, nullable = false)
    private String id;

    @Column(name="email",length = 255, nullable = false)
    private String email;

    @Column(name="pw",length = 255, nullable = false)
    private String pw;

    @Column(name="nick",length = 50, nullable = false)
    private String nick;

    @Column(name="birthday",nullable = false)
    private LocalDate birthday;

    @Column(name="gender",columnDefinition = "varchar(2) check (gender in ('m','f'))" ,nullable = false)
    private String gender;

    @Column(columnDefinition = " varchar(5) default '회원'")
    private int status;


    public void changePw(String user_pw){
        this.pw = user_pw;
    }

    public void changeNick(String nick){
        this.nick = nick;
    }

    public void changeGender(String gender){
        this.gender = gender;
    }

    public void changeStatus(int status) {
        this.status = status;
    }

    public void changeBirthday(LocalDate birthday) {
        this.birthday = birthday;
    }

}
```

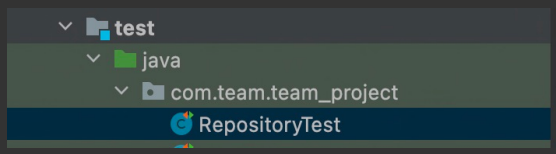
repository package 를 생성하고
UserRepository interface 를 생성



UserRepository.java

```java
package com.team.team_project.repository;

import com.team.team_project.entity.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User,Long> {
}
```

데이터 삽입 테스트

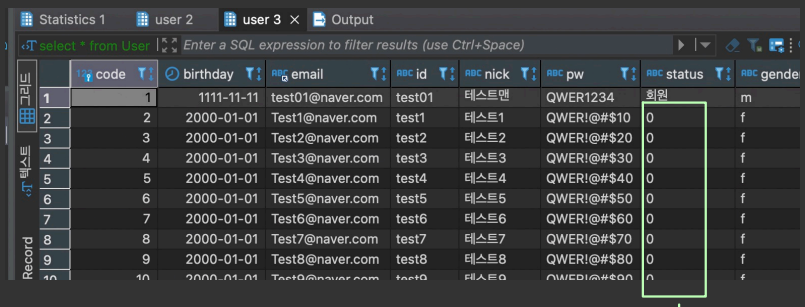


RepositoryTest.java

```java
package com.team.team_project;

import com.team.team_project.entity.User;
import com.team.team_project.repository.UserRepository;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

import java.time.LocalDate;
import java.util.stream.IntStream;

@SpringBootTest
public class RepositoryTest {

    @Autowired
    private UserRepository userRepository;

    @Test
    public void userTest(){
        IntStream.rangeClosed(1,100).forEach(i->{
            User user = User.builder()
                    .email("Test"+i+"@naver.com")
                    .id("test"+i)
                    .nick("테스트"+i)
                    .pw("QWER!@#$"+i*10)
                    .birthday(LocalDate.parse("2000-01-01"))
                    .gender("f")
                    .build();
            userRepository.save(user);
        });

    }
}
```

Result

```
        user
        (birthday, email, gender, id, nick, pw, status)
    values
        (?, ?, ?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        user
        (birthday, email, gender, id, nick, pw, status)
    values
        (?, ?, ?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        user
        (birthday, email, gender, id, nick, pw, status)
    values
        (?, ?, ?, ?, ?, ?, ?)
```



Statistics 1 | user 2 | user 3 | Output

select * from User — Enter a SQL expression to filter results (use Ctrl+Space)

	code	birthday	email	id	nick	pw	status	gende
1	1	1111-11-11	test01@naver.com	test01	테스트맨	QWER1234	회원	m
2	2	2000-01-01	Test1@naver.com	test1	테스트1	QWER!@#$10	0	f
3	3	2000-01-01	Test2@naver.com	test2	테스트2	QWER!@#$20	0	f
4	4	2000-01-01	Test3@naver.com	test3	테스트3	QWER!@#$30	0	f
5	5	2000-01-01	Test4@naver.com	test4	테스트4	QWER!@#$40	0	f
6	6	2000-01-01	Test5@naver.com	test5	테스트5	QWER!@#$50	0	f
7	7	2000-01-01	Test6@naver.com	test6	테스트6	QWER!@#$60	0	f
8	8	2000-01-01	Test7@naver.com	test7	테스트7	QWER!@#$70	0	f
9	9	2000-01-01	Test8@naver.com	test8	테스트8	QWER!@#$80	0	f

Default 일 때 회원이 나와야 한다.

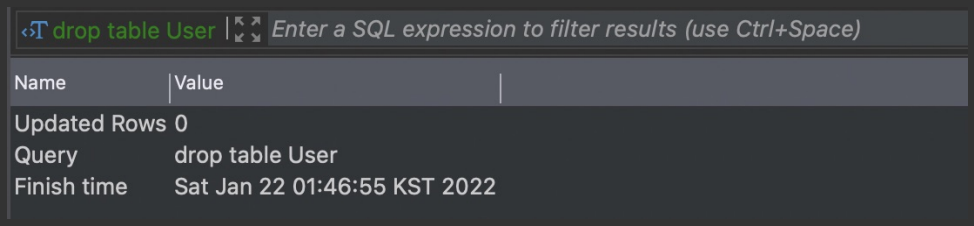
원인 : Entity 에서 status 의 데이터 타입이 int 로 되어 있었음

User.java

```
... 생략

@Column(columnDefinition = " varchar(5) default '회원'")
private String status;

... 생략
public void changeStatus(String status) {
    this.status = status;
}
```

MySQL 에서 Drop table 을 통해
—> Table 삭제 후 다시 실행

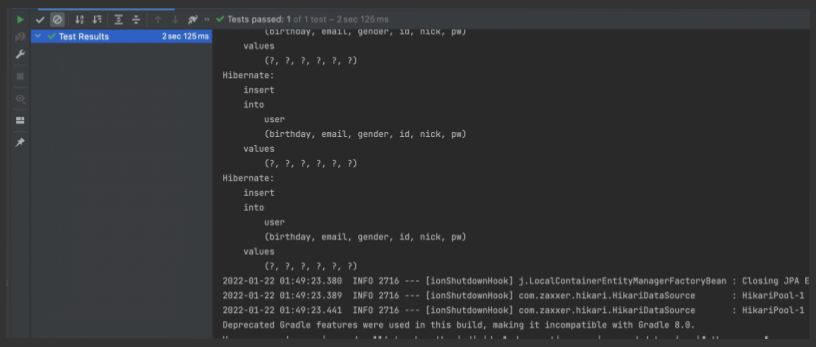
```
drop table User;
```



RepositoryTest.java

```
@Test
    public void userTest(){
        IntStream.rangeClosed(1,100).forEach(i->{
            User user = User.builder()
                    .email("Test"+i+"@naver.com")
                    .id("test"+i)
                    .nick("테스트"+i)
                    .pw("QWER!@#$"+i*10)
                    .birthday(LocalDate.parse("2000-01-01"
))
                    .gender("f")
                    .build();
            userRepository.save(user);
        });
    }
}
```

OPERTATE RESULT -1



OPERTATE RESULT -2

default 값으로 '회원' 넣는 것은 처리

MySQL.TEAM(Database)

select * from User | Enter a SQL expression to filter results (use Ctrl+Space)

	code	moddate	regdate	birthday	email	gender	id	nick	pw	status
1	1	[NULL]	[NULL]	2000-01-01	Test1@naver.com	f	test1	테스트1	QWER!@#$10	회원
2	2	[NULL]	[NULL]	2000-01-01	Test2@naver.com	f	test2	테스트2	QWER!@#$20	회원
3	3	[NULL]	[NULL]	2000-01-01	Test3@naver.com	f	test3	테스트3	QWER!@#$30	회원
4	4	[NULL]	[NULL]	2000-01-01	Test4@naver.com	f	test4	테스트4	QWER!@#$40	회원
5	5	[NULL]	[NULL]	2000-01-01	Test5@naver.com	f	test5	테스트5	QWER!@#$50	회원
6	6	[NULL]	[NULL]	2000-01-01	Test6@naver.com	f	test6	테스트6	QWER!@#$60	회원
7	7	[NULL]	[NULL]	2000-01-01	Test7@naver.com	f	test7	테스트7	QWER!@#$70	회원

regdate 와 modate 설정이 안되어있음 .

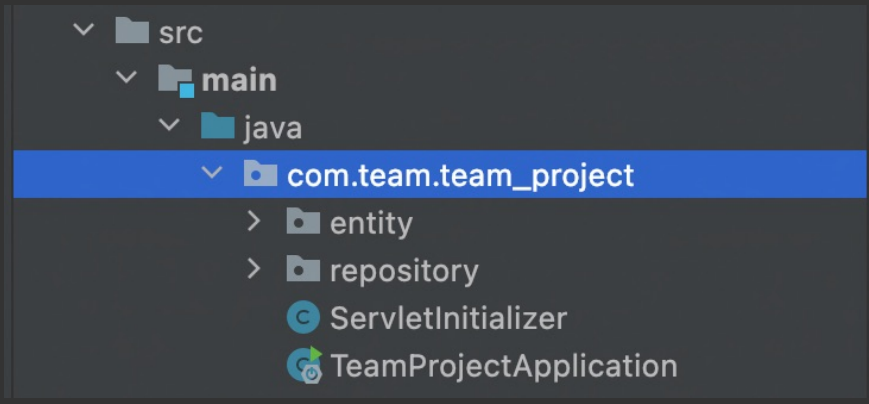
등록 날짜 (regDate) 수정 날짜 (modDate) 수정 하기

원인 추정

——〉
src/main/java/기본 package/ 의 Application 파일에

자동으로 날짜와 시간 처리를 해주는

@EnableJpaAuditing
추가하지 않았다



TeamProjectApplicaiton.java

```java
package com.team.team_project;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.data.jpa.repository.config.EnableJpaAuditing;


//자동으로 날짜와 시간 처리
@EnableJpaAuditing
@SpringBootApplication
public class TeamProjectApplication {

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

}
```

TEST

다시 Table 을 Drop 하고
RepositoryTest.java 에서 method 를 실행

TEST-RESULT

user 1 ×

select * from User | Enter a SQL expression to filter results (use Ctrl+Space)

그리드 | 텍스트 | Record

	code	moddate	regdate	birthday	email	gender	id	nick	pw	status
1	1	2022-01-22 02:07:32.729252000	2022-01-22 02:07:32.729252000	2000-01-01	Test1@naver.com	f	test1	테스트1	QWER!@#$10	회원
2	2	2022-01-22 02:07:32.778758000	2022-01-22 02:07:32.778758000	2000-01-01	Test2@naver.com	f	test2	테스트2	QWER!@#$20	회원
3	3	2022-01-22 02:07:32.784298000	2022-01-22 02:07:32.784298000	2000-01-01	Test3@naver.com	f	test3	테스트3	QWER!@#$30	회원
4	4	2022-01-22 02:07:32.790344000	2022-01-22 02:07:32.790344000	2000-01-01	Test4@naver.com	f	test4	테스트4	QWER!@#$40	회원
5	5	2022-01-22 02:07:32.795657000	2022-01-22 02:07:32.795657000	2000-01-01	Test5@naver.com	f	test5	테스트5	QWER!@#$50	회원
6	6	2022-01-22 02:07:32.802124000	2022-01-22 02:07:32.802124000	2000-01-01	Test6@naver.com	f	test6	테스트6	QWER!@#$60	회원
7	7	2022-01-22 02:07:32.807848000	2022-01-22 02:07:32.807848000	2000-01-01	Test7@naver.com	f	test7	테스트7	QWER!@#$70	회원
8	8	2022-01-22 02:07:32.814576000	2022-01-22 02:07:32.814576000	2000-01-01	Test8@naver.com	f	test8	테스트8	QWER!@#$80	회원
9	9	2022-01-22 02:07:32.823090000	2022-01-22 02:07:32.823090000	2000-01-01	Test9@naver.com	f	test9	테스트9	QWER!@#$90	회원
10	10	2022-01-22 02:07:32.830431000	2022-01-22 02:07:32.830431000	2000-01-01	Test10@naver.com	f	test10	테스트10	QWER!@#$100	회원
11	11	2022-01-22 02:07:32.834745000	2022-01-22 02:07:32.834745000	2000-01-01	Test11@naver.com	f	test11	테스트11	QWER!@#$110	회원

———————— 기본적인 Entity 생성과 , 데이터 삽입 TEST 완료

Entity

User.java

```java
package com.team.team_project.entity;

import lombok.*;
import org.hibernate.annotations.DynamicInsert;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import javax.persistence.*;
import java.time.LocalDate;

@Entity
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Getter
@ToString
@EntityListeners(value={AuditingEntityListener.class})
@Table(name="User",uniqueConstraints = {
        @UniqueConstraint(columnNames = {"email","id","nick"})})
@DynamicInsert
public class User extends BaseEntity{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long code;

    @Column(name="id", length = 255, nullable = false)
    private String id;

    @Column(name="email",length = 255, nullable = false)
    private String email;

    @Column(name="pw",length = 255, nullable = false)
    private String pw;

    @Column(name="nick",length = 50, nullable = false)
    private String nick;

    @Column(name="birthday",nullable = false)
    private LocalDate birthday;

    @Column(name="gender",columnDefinition = "varchar(2) check (gender in ('m','f'))" ,nullable = false)
    private String gender;

    @Column(columnDefinition = " varchar(5) default '회원'")
    private String status;


    public void changePw(String user_pw){
        this.pw = user_pw;
    }

    public void changeNick(String nick){
        this.nick = nick;
    }

    public void changeGender(String gender){
        this.gender = gender;
    }

    public void changeStatus(String status) {
        this.status = status;
    }

    public void changeBirthday(LocalDate birthday) {
        this.birthday = birthday;
    }

}
```

BaseEntity.java

```java
package com.team.team_project.entity;

import lombok.Getter;
import org.springframework.data.annotation.CreatedDate;
import org.springframework.data.annotation.LastModifiedDate;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import javax.persistence.Column;
import javax.persistence.EntityListeners;
import javax.persistence.MappedSuperclass;
import java.time.LocalDateTime;

@MappedSuperclass
@EntityListeners(value = {AuditingEntityListener.class})
@Getter
public class BaseEntity {
    @CreatedDate
    @Column(name="regdate", updatable = false)
    private LocalDateTime regDate;

    @LastModifiedDate
    @Column(name="moddate")     //자동 업데이트
    private LocalDateTime modDate;
}
```

Application

TeamProjectApplication.java

```java
package com.team.team_project;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.data.jpa.repository.config.EnableJpaAuditing;


//자동으로 날짜와 시간 처리
@EnableJpaAuditing
@SpringBootApplication
public class TeamProjectApplication {

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

Repository

UserRepository.java

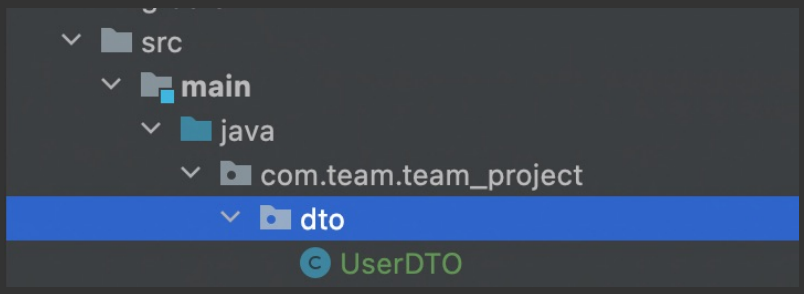
```java
package com.team.team_project.repository;

import com.team.team_project.entity.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User,Long> {
}
```

TEST

RepositoryTest.java

```java
package com.team.team_project;

import com.team.team_project.entity.User;
import com.team.team_project.repository.UserRepository;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

import java.time.LocalDate;
import java.util.stream.IntStream;

@SpringBootTest
public class RepositoryTest {

    @Autowired
    private UserRepository userRepository;

    @Test
    public void userTest(){
        IntStream.rangeClosed(1,100).forEach(i->{
            User user = User.builder()
                    .email("Test"+i+"@naver.com")
                    .id("test"+i)
                    .nick("테스트"+i)
                    .pw("QWER!@#$"+i*10)
                    .birthday(LocalDate.parse("2000-01-01"))
                    .gender("f")
                    .build();
            userRepository.save(user);
        });
    }
}
```

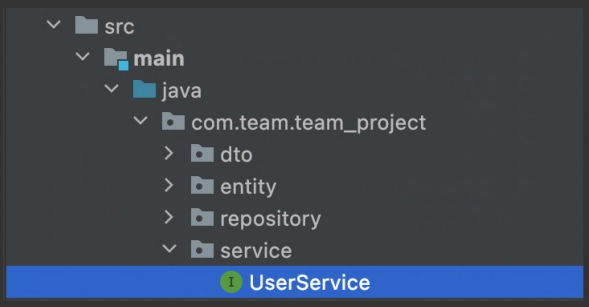
DTO 만들기



UserDTO.java

```java
package com.team.team_project.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDateTime;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class UserDTO {
    private Long code;
    private String id;
    private String pw;
    private String email;
    private String nick;
    private String gender;
    private String birthday;
    private String status;
    private LocalDateTime regDate;
}
```

Entity 와 DTO 연결하기
—→
service package 생성하고 Interface 생성



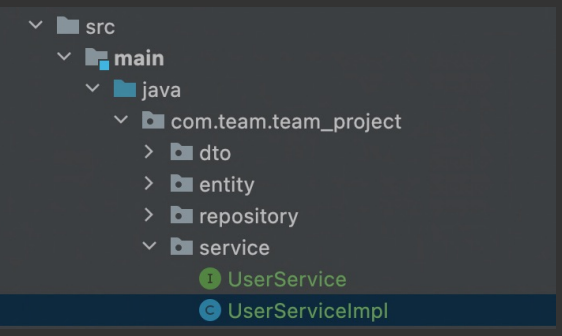
UserService.java

```java
package com.team.team_project.service;

import com.team.team_project.dto.UserDTO;
import com.team.team_project.entity.User;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public interface UserService {
    // default 를 사용하는 이유
    // interface 내에서도  로직이 포함된 메소드를 선언할 수 있게 하기 위해서 .
    // (하위 호환성 때문임)


    // DTO 를 Entity 로 변환
    default User dtoToEntity(UserDTO dto){
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        LocalDate date = LocalDate.parse(dto.getBirthday(), formatter);
        User entity = User.builder()
                .code(dto.getCode())
                .id(dto.getId())
                .pw(dto.getPw())
                .email(dto.getEmail())
                .nick(dto.getNick())
                .gender(dto.getGender())
                .birthday(date)
                .status(dto.getStatus())
                .build();
        return entity;
    }

    // Entity 를 DTO 로 변경
    default UserDTO entityToDto(User entity) {
        UserDTO dto = UserDTO.builder()
                .code(entity.getCode())
                .id(entity.getId())
                .pw(entity.getPw())
                .email(entity.getEmail())
                .nick(entity.getNick())
                .gender(entity.getGender())
                .birthday(String.valueOf(entity.getBirthday()))
                .status(entity.getStatus())
                .regDate(entity.getRegDate())
                .build();
        return dto;
    }


    // 회원 가입을 위한 method 생성
    public Long join(UserDTO dto);
}
```

return type 을 Long 으로 지정

Service Package 에 Service 를 구현 시킬 ServiceImpl 파일 생성
——〉 UserService 파일을 implements 받음

UserServiceImpl



UserServiceImpl.java

```java
package com.team.team_project.service;

import com.team.team_project.dto.UserDTO;
import com.team.team_project.entity.User;
import com.team.team_project.repository.UserRepository;
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.springframework.stereotype.Service;

@Log4j2
@Service
@RequiredArgsConstructor
public class UserServiceImpl implements UserService{

    private final UserRepository userRepository;

    @Override
    public Long join(UserDTO dto) {
        User entity = dtoToEntity(dto);
        userRepository.save(entity);
        return entity.getCode();
    }
}
```

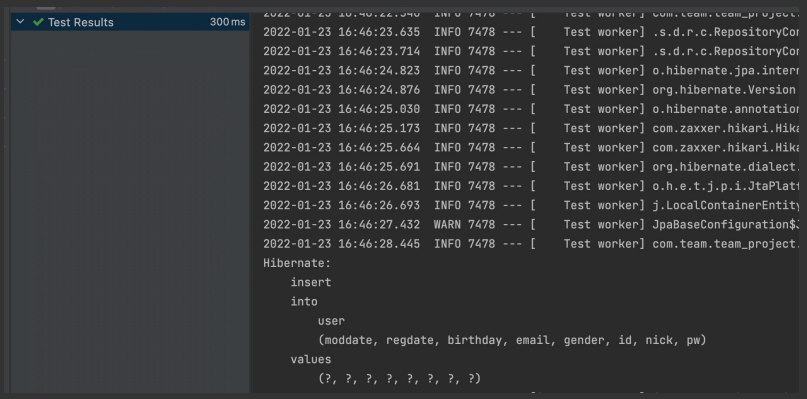
TEST ——〉

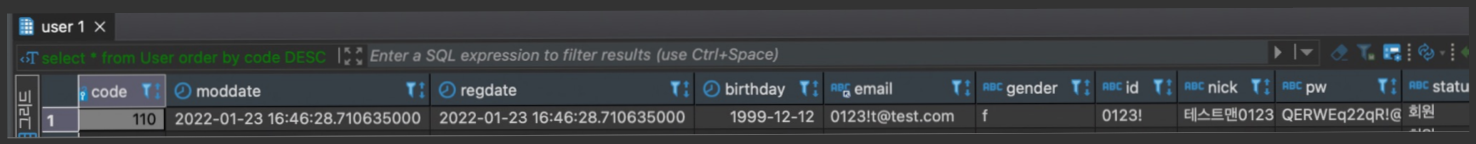
DTO 를 통해서 DATA base 에 데이터가 삽입되는 것을 목표로 함

RepositoryTest.java

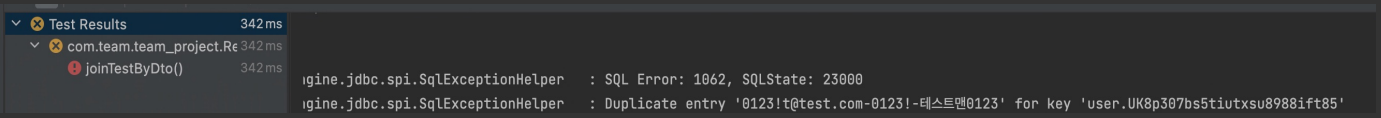
```java
@Autowired
    private UserService userService;

    @Test
    public void joinTestByDto(){
        UserDTO dto = UserDTO.builder()
                .id("0123!")
                .email("0123!t@test.com")
                .nick("테스트맨0123")
                .pw("QERWEq22qR!@#")
                .birthday("1999-12-12")
                .gender("f")
                .build();
        userService.join(dto);
    }
}
```

TEST - RESULT





한번더 실행하면 unique로 설정된 값들 때문에 Duplicate Error 발생



—— Annotation 정리.

@Builder

Boilerplate Code를 줄일 수 있다.
(반드시 필요한 코드지만 반복적으로 사용되는 코드)
자동으로 해당 클래스에 빌더를 추가해주기 때문에 매우 편리

```
UserDTO dto = UserDTO.builder()
        .id("0123!")
        .email("0123!t@test.com")
        .nick("테스트맨0123")
        .pw("QERWEq22qR!@#")
        .birthday("1999-12-12")
        .gender("f")
        .build();
```

이런 거 생성 가능

Builder Pattern

인스턴스 생성시 생성자로만 생성하는 데에 어려움이 있다.

그 문제를 해결하기 위해서

점층적 생성자 패턴(telescoping constructor pattern)
: 동일한 Class 에 Overloading 을 이용하여 생성자를 여러개 생성

과

자바 빈 패턴(Java Bean Pattern)
: setter method 를 사용

을 결합한 형태

: 필수 인자들을 전달하여 빌더 객체를 만들고 ,
빌더 객체에서 정의된 설정 method 들을
호출하여 인스턴스들을 생성

: 장점
1. 인스턴스 생성시 , Essential 과 selectial 을 구분가능

2. 객체 일관성을 깨뜨리지 않을 수 있다 .

@AllArgsConstructor

어노테이션은 모든 필드 값을 파라미터로 받는 생성자를 생성

@NoArgsConstructor

어노테이션은 파라미터가 없는 기본 생성자를 생성

@RequiredArgsConstructor

final이나 @NonNull인 필드 값만 파라미터로 받는 생성자를 생성

@EntityListeners

JPA Entity에 Event 발생 시 콜백을 처리하고 코드를 실행하는 방법
Persist, Remove, Update, Load에 대한 event 전과 후에 대한 콜백 메서드를 제공

DTO

UserDTO.java

```java
package com.team.team_project.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDateTime;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class UserDTO {
    private Long code;
    private String id;
    private String pw;
    private String email;
    private String nick;
    private String gender;
    private String birthday;
    private String status;
    private LocalDateTime regDate;
}
```

Service

UserService.java(interface)

```java
package com.team.team_project.service;

import com.team.team_project.dto.UserDTO;
import com.team.team_project.entity.User;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public interface UserService {
    // default 를 사용하는 이유
    // interface 내에서도  로직이 포함된 메소드를 선언할 수 있게 하기 위해서 .
    // (하위 호환성 때문임)


    // DTO 를 Entity 로 변환
    default User dtoToEntity(UserDTO dto){
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        LocalDate date = LocalDate.parse(dto.getBirthday(), formatter);
        User entity = User.builder()
                .code(dto.getCode())
                .id(dto.getId())
                .pw(dto.getPw())
                .email(dto.getEmail())
                .nick(dto.getNick())
                .gender(dto.getGender())
                .birthday(date)
                .status(dto.getStatus())
                .build();
        return entity;
    }

    // Entity 를 DTO 로 변경
    default UserDTO entityToDto(User entity) {
        UserDTO dto = UserDTO.builder()
                .code(entity.getCode())
                .id(entity.getId())
                .pw(entity.getPw())
                .email(entity.getEmail())
                .nick(entity.getNick())
                .gender(entity.getGender())
                .birthday(String.valueOf(entity.getBirthday()))
                .status(entity.getStatus())
                .regDate(entity.getRegDate())
                .build();
        return dto;
    }


    // 회원 가입을 위한 method 생성
    public Long join(UserDTO dto);
}
```

ServiceImpl

UserServiceImpl.java

```java
package com.team.team_project.service;

import com.team.team_project.dto.UserDTO;
import com.team.team_project.entity.User;
import com.team.team_project.repository.UserRepository;
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.springframework.stereotype.Service;

@Log4j2
@Service
@RequiredArgsConstructor
public class UserServiceImpl implements UserService{

    private final UserRepository userRepository;

    @Override
    public Long join(UserDTO dto) {
        User entity = dtoToEntity(dto);
        userRepository.save(entity);
        return entity.getCode();
    }
}
```

TEST

RepositoryTest.java

```java
package com.team.team_project;

import com.team.team_project.dto.UserDTO;
import com.team.team_project.entity.User;
import com.team.team_project.repository.UserRepository;
import com.team.team_project.service.UserService;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

import java.time.LocalDate;
import java.util.stream.IntStream;

@SpringBootTest
public class RepositoryTest {

    @Autowired
    private UserRepository userRepository;

//    @Test
    public void userTest(){
        IntStream.rangeClosed(1,100).forEach(i->{
            User user = User.builder()
                    .email("Test"+i+"@naver.com")
                    .id("test"+i)
                    .nick("테스트"+i)
                    .pw("QWER!@#$"+i*10)
                    .birthday(LocalDate.parse("2000-01-01"))
                    .gender("f")
                    .build();
            userRepository.save(user);
        });
    }

    @Autowired
    private UserService userService;

    @Test
    public void joinTestByDto(){
        UserDTO dto = UserDTO.builder()
                .id("0123!_1")
                .email("0123!_1t@test.com")
                .nick("테스트맨0123_1")
                .pw("QERWEq22qR!@#")
                .birthday("1999-12-12")
                .gender("f")
                .build();
        userService.join(dto);
    }
}
```

PLAN Table Entity 만들기

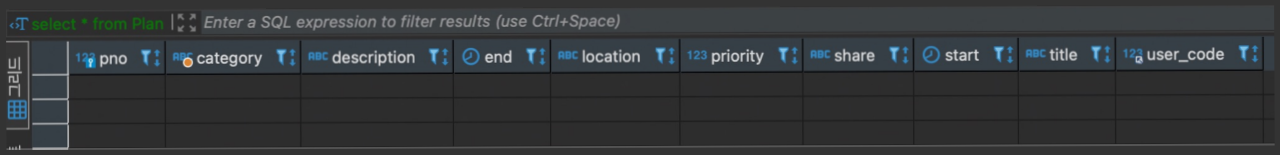
SQL

```sql
create table Plan(
    code int not null auto_increment,
    user int,
    priority int ,
    title varchar(100) not null,
    location varchar(100),
    start date not null,
    end date not null,
    description text,
    category char(20) not null default 'No Category',
    share char(4) check(share in ('공개','비공개')) default '비공개',
    constraint PK_Plan primary key (code),
    constraint User_FK_Plan FOREIGN KEY (user) REFERENCES User(code) on delete cascade
)DEFAULT CHARSET=utf8;
```

Plan.java

```java
package com.team.team_project.entity;

import lombok.*;

import javax.persistence.*;
import java.time.LocalDateTime;

@Entity
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Getter
@ToString
public class Plan extends BaseEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long pno;

    private Long priority;

    @Column(length = 100 , nullable = false)
    private String title;

    private String location;

    private String description;

    @Column(nullable = false)
    private LocalDateTime start;

    @Column(nullable = false)
    private LocalDateTime end;

    @Column(length = 20 , columnDefinition = "varchar(20) default 'No Category'",nullable = false)
    private String category;

    @Column(length= 20 , columnDefinition = "varchar(20) check (share in ('공개','비공개')) default '비공개'")
    private String share;


    @ManyToOne(fetch = FetchType.LAZY)
    private User user;
}
```

RESULT



MYSQL

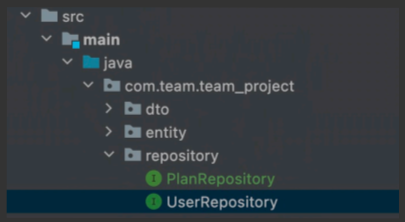
```sql
CREATE TABLE `plan` (
  `pno` bigint NOT NULL AUTO_INCREMENT,
  `moddate` datetime(6) DEFAULT NULL,
  `regdate` datetime(6) DEFAULT NULL,
  `category` varchar(20) NOT NULL DEFAULT 'No Category',
  `description` varchar(255) DEFAULT NULL,
  `end` datetime(6) NOT NULL,
  `location` varchar(255) DEFAULT NULL,
  `priority` bigint DEFAULT NULL,
  `share` varchar(20) DEFAULT '비공개',
  `start` datetime(6) NOT NULL,
  `title` varchar(100) NOT NULL,
  `user_code` bigint DEFAULT NULL,
  PRIMARY KEY (`pno`),
  KEY `FKm46h77ilxf1ji9k1qs22fgavl` (`user_code`),
  CONSTRAINT `FKm46h77ilxf1ji9k1qs22fgavl` FOREIGN KEY (`user_code`) REFERENCES `user` (`code`),
  CONSTRAINT `plan_chk_1` CHECK ((`share` in (_utf8mb4'공개',_utf8mb4'비공개')))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

추후에 변경될 수 있는 값 들을 지정해서 method 생성
// ToString 에서 exclude 속성

Plan.java

```java
package com.team.team_project.entity;

import lombok.*;

import javax.persistence.*;
import java.time.LocalDate;
import java.time.LocalDateTime;

@Entity
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Getter
@ToString(exclude="user")
// 특정 필드 "user" 를 결과에서 제외시키는 거 가능
public class Plan extends BaseEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long pno;

    private Long priority;

    @Column(length = 100 , nullable = false)
    private String title;

    private String location;

    private String description;

    @Column(nullable = false)
    private LocalDateTime start;

    @Column(nullable = false)
    private LocalDateTime end;

    @Column(length = 20 , columnDefinition = "varchar(20) default 'No Category'",nullable = false)
    private String category;

    @Column(length= 20 , columnDefinition = "varchar(20) check (share in ('공개','비공개')) default '비공개'")
    private String share;


    @ManyToOne(fetch = FetchType.LAZY)
    private User user;

    public void changePriority(Long priority){
        this.priority=priority;
    }
    public void  changeTitle(String title){
        this.title=title;
    }
    public void changeLocation(String location){
        this.location=location;
    }
    public void changeDescription(String description){
        this.description=description;
    }
    public void changeStart(LocalDate start){
        this.start= LocalDateTime.from(start);
    }
    public void changeEnd(LocalDate end){
        this.end= LocalDateTime.from(end);
    }
    public void changeShare(String share){
        this.share=share;
    }
    public void changeCategory(String category){
        this.category=category;
    }
}
```

PlanRepository 생성



PlanRepository.java(interface)

```java
package com.team.team_project.repository;

import com.team.team_project.entity.Plan;
import org.springframework.data.jpa.repository.JpaRepository;

public interface PlanRepository extends JpaRepository<Plan,Long> {
}
```

PlanDTO 생성

PlanDTO.java

```java
package com.team.team_project.dto;


import lombok.*;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
@ToString
public class PlanDTO {
    private Long pno;
    private Long priority;
    private String title;
    private String location;
    private String description;
    private String start;
    private String end;
    private String category;
    private String share;
    private Long user_code;
}
```

Service Class 생성
————〉 Entity 와 DTO 연결

PlanService.java(interface)

```
package com.team.team_project.service;

import com.team.team_project.dto.PlanDTO;
import com.team.team_project.entity.Plan;
import com.team.team_project.entity.User;

public interface PlanService {

    default Plan dtoToEntity(PlanDTO dto){
        User user = User.builder()
                .code(dto.getUser_code())
                .build();
        Plan plan = Plan.builder()
                .pno(dto.getPno())
                .priority(dto.getPriority())
                .title(dto.getTitle())
                .category(dto.getCategory())
                .description(dto.getDescription())
                .share(dto.getShare())
                .location(dto.getLocation())
                .start(dto.getStart())
                .end(dto.getEnd())
                .user(user)
                .build();
        return plan;
    }

    default PlanDTO entityToDTO(Plan plan,User user){
        PlanDTO dto =PlanDTO.builder()
                .user_code(user.getCode())
                .pno(plan.getPno())
                .priority(plan.getPriority())
                .title(plan.getTitle())
                .description(plan.getDescription())
                .location(plan.getLocation())
                .category(plan.getCategory())
                .share(plan.getShare())
                .start(plan.getStart())
                .end(plan.getEnd())
                .build();
        return dto;
    }

     Long makeAPlan(PlanDTO dto);
}
```

ServiceImpl file 생성

PlanServiceImpl.java

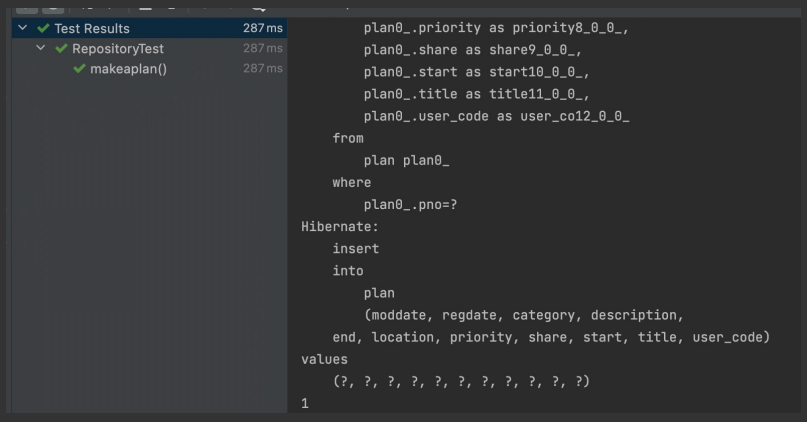
```java
package com.team.team_project.service;


import com.team.team_project.dto.PlanDTO;
import com.team.team_project.entity.Plan;
import com.team.team_project.repository.PlanRepository;
import lombok.NoArgsConstructor;
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
@RequiredArgsConstructor

@Log4j2
public class PlanServiceImpl implements PlanService {
    private final PlanRepository planRepository;


    @Override
    public Long makeAPlan(PlanDTO dto) {
        Plan entity = dtoToEntity(dto);
        planRepository.save(entity);
        return entity.getPno();
    }
}
```

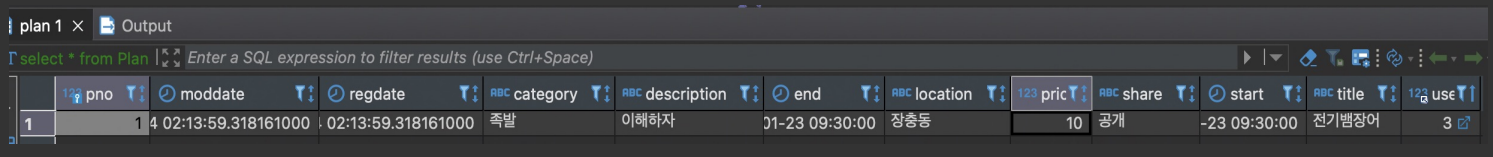
TEST

RepositoryTest.java

```java
@Autowired
private PlanService planService;

@Test
public void makeaplan(){
    PlanDTO dto =PlanDTO.builder()
            .user_code(3L)
            .pno(1L)
            .priority(10L)
            .title("전기뱀장어")
            .description("이해하자")
            .location("장충동")
            .category("족발")
            .share("공개")
            .start(LocalDateTime.parse("2002-01-23T09:30:00"))
            .end(LocalDateTime.parse("2002-01-23T09:30:00"))
            .build();
    System.out.println(planService.makeAPlan(dto));
```

TESTRESULT





데이터 100 개 넣기 2 번

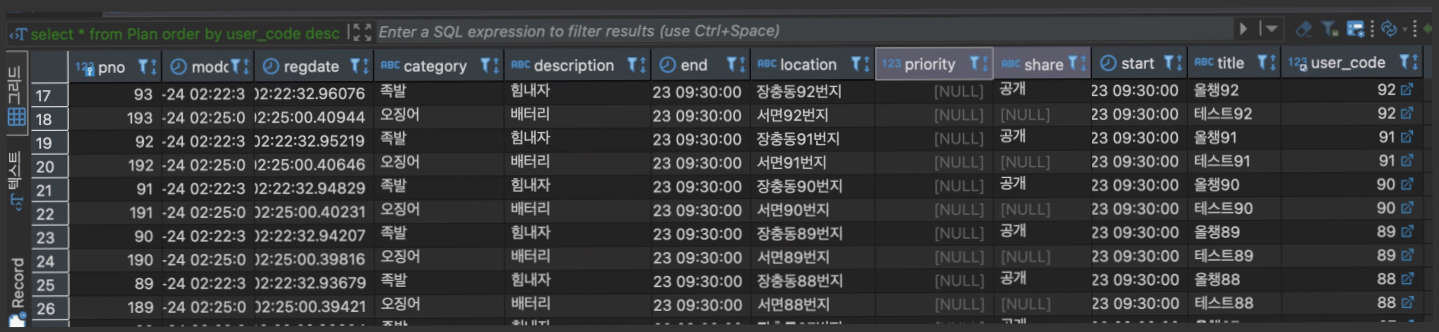
RepositoryTest.java

```
@Test
public void hundredmakeplan(){
    IntStream.rangeClosed(1,100).forEach(i->{
        PlanDTO dto =PlanDTO.builder()
                .user_code((long)i)
                .title("올챙"+i)
                .description("힘내자")
                .location("장충동"+i+"번지")
                .category("족발")
                .share("공개")
                .start(LocalDateTime.parse("2002-01-23T09:30:00"))
                .end(LocalDateTime.parse("2002-01-23T09:30:00"))
                .build();
        System.out.println(planService.makeAPlan(dto));
    });
}
```

RepositoryTest.java

```
@Test
public void hundredmakeplan(){
    IntStream.rangeClosed(1,100).forEach(i->{
        PlanDTO dto =PlanDTO.builder()
                .user_code((long)i)
                .title("테스트"+i)
                .description("배터리")
                .location("서면"+i+"번지")
                .category("오징어")
                .start(LocalDateTime.parse("2002-01-23T09:30:00"))
                .end(LocalDateTime.parse("2002-01-23T09:30:00"))
                .build();
        System.out.println(planService.makeAPlan(dto));
    });
}
```

TEST _RESULT:



select * from Plan order by user_code desc

Enter a SQL expression to filter results (use Ctrl+Space)

	pno	modc	regdate	category	description	end	location	priority	share	start	title	user_code
17	93	-24 02:22:3	)2:22:32.96076	족발	힘내자	23 09:30:00	장충동92번지	[NULL]	공개	23 09:30:00	올챙92	92
18	193	-24 02:25:0	)2:25:00.40944	오징어	배터리	23 09:30:00	서면92번지	[NULL]	[NULL]	23 09:30:00	테스트92	92
19	92	-24 02:22:3	)2:22:32.95219	족발	힘내자	23 09:30:00	장충동91번지	[NULL]	공개	23 09:30:00	올챙91	91
20	192	-24 02:25:0	)2:25:00.40646	오징어	배터리	23 09:30:00	서면91번지	[NULL]	[NULL]	23 09:30:00	테스트91	91
21	91	-24 02:22:3	)2:22:32.94829	족발	힘내자	23 09:30:00	장충동90번지	[NULL]	공개	23 09:30:00	올챙90	90
22	191	-24 02:25:0	)2:25:00.40231	오징어	배터리	23 09:30:00	서면90번지	[NULL]	[NULL]	23 09:30:00	테스트90	90
23	90	-24 02:22:3	)2:22:32.94207	족발	힘내자	23 09:30:00	장충동89번지	[NULL]	공개	23 09:30:00	올챙89	89
24	190	-24 02:25:0	)2:25:00.39816	오징어	배터리	23 09:30:00	서면89번지	[NULL]	[NULL]	23 09:30:00	테스트89	89
25	89	-24 02:22:3	)2:22:32.93679	족발	힘내자	23 09:30:00	장충동88번지	[NULL]	공개	23 09:30:00	올챙88	88
26	189	-24 02:25:0	)2:25:00.39421	오징어	배터리	23 09:30:00	서면88번지	[NULL]	[NULL]	23 09:30:00	테스트88	88

——————〉 Entity 에서 Share default 값 설정이 안먹었다.

원인 DynamicInsert Annotation 을 Entity 에서 선언을 하지 않음 .

Plan.java(Entity)

```java
package com.team.team_project.entity;

import lombok.*;
import org.hibernate.annotations.DynamicInsert;

import javax.persistence.*;
import java.time.LocalDate;
import java.time.LocalDateTime;

@Entity
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Getter
@ToString(exclude="user")
@DynamicInsert
// 특정 필드 "user" 를 결과에서 제외시키는 거 가능
public class Plan extends BaseEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long pno;

    private Long priority;

    @Column(length = 100 , nullable = false)
    private String title;

    private String location;

    private String description;

    @Column(nullable = false)
    private LocalDateTime start;

    @Column(nullable = false)
    private LocalDateTime end;

    @Column(length = 20 , columnDefinition = "varchar(20) default 'No Category'")
    private String category;

    @Column(length= 20 , columnDefinition = "varchar(20) check (share in ('공개','비공개')) default '비공개'")
    private String share;


    @ManyToOne(fetch = FetchType.LAZY)
    private User user;

    public void changePriority(Long priority){
        this.priority=priority;
    }
    public void  changeTitle(String title){
        this.title=title;
    }
    public void changeLocation(String location){
        this.location=location;
    }
    public void changeDescription(String description){
        this.description=description;
    }
    public void changeStart(LocalDate start){
        this.start= LocalDateTime.from(start);
    }
    public void changeEnd(LocalDate end){
        this.end= LocalDateTime.from(end);
    }
    public void changeShare(String share){
        this.share=share;
    }
    public void changeCategory(String category){
        this.category=category;
    }
}
```

NotNull 제외

TEST — 기존의 Table 지우고 실행

RepositoryTest.java

```
@Test
    public void hundredmakeplan(){
        IntStream.rangeClosed(1,100).forEach(i->{
            PlanDTO dto =PlanDTO.builder()
                    .user_code((long)i)
                    .title("테스트"+i)
                    .description("TEST_1"+i)
                    .location("TEST"+i+"번지")
                    .start(LocalDateTime.parse("2002-01-23T09:30:00"))
                    .end(LocalDateTime.parse("2002-01-23T09:30:00"))
                    .build();
            System.out.println(planService.makeAPlan(dto));
        });
    }
}
```

TEST_Result

select * from Plan order by user_code desc | Enter a SQL expression to filter results (use Ctrl+Space)

	ate	category	description	end	location	priority	share	start	title	user_code
1	1-24 02:34:49.482571000	[NULL]	배터리	2002-01-23 09:30:00	서면100번지	[NULL]	[NULL]	2002-01-23 09:30:00	테스트100	100
2	1-24 02:38:38.601074000	No Category	TEST_1100	2002-01-23 09:30:00	TEST100번지	[NULL]	비공개	2002-01-23 09:30:00	테스트100	100
3	1-24 02:34:49.479391000	[NULL]	배터리	2002-01-23 09:30:00	서면99번지	[NULL]	[NULL]	2002-01-23 09:30:00	테스트99	99
4	1-24 02:38:38.594468000	No Category	TEST_199	2002-01-23 09:30:00	TEST99번지	[NULL]	비공개	2002-01-23 09:30:00	테스트99	99
5	1-24 02:34:49.469966000	[NULL]	배터리	2002-01-23 09:30:00	서면98번지	[NULL]	[NULL]	2002-01-23 09:30:00	테스트98	98
6	1-24 02:38:38.584107000	No Category	TEST_198	2002-01-23 09:30:00	TEST98번지	[NULL]	비공개	2002-01-23 09:30:00	테스트98	98
7	1-24 02:34:49.464220000	[NULL]	배터리	2002-01-23 09:30:00	서면97번지	[NULL]	[NULL]	2002-01-23 09:30:00	테스트97	97
8	1-24 02:38:38.563475000	No Category	TEST_197	2002-01-23 09:30:00	TEST97번지	[NULL]	비공개	2002-01-23 09:30:00	테스트97	97
9	1-24 02:34:49.458716000	[NULL]	배터리	2002-01-23 09:30:00	서면96번지	[NULL]	[NULL]	2002-01-23 09:30:00	테스트96	96

DTO를 통해 PlanEntity 로 Database에 data 삽입 완료

Entity

Plan.java

```java
package com.team.team_project.entity;

import lombok.*;
import org.hibernate.annotations.DynamicInsert;

import javax.persistence.*;
import java.time.LocalDate;
import java.time.LocalDateTime;

@Entity
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Getter
@ToString(exclude="user")
@DynamicInsert
// 특정 필드 "user" 를 결과에서 제외시키는 거 가능
public class Plan extends BaseEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long pno;

    private Long priority;

    @Column(length = 100 , nullable = false)
    private String title;

    private String location;

    private String description;

    @Column(nullable = false)
    private LocalDateTime start;

    @Column(nullable = false)
    private LocalDateTime end;

    @Column(length = 20 , columnDefinition = "varchar(20) default 'No Category'")
    private String category;

    @Column(length= 20 , columnDefinition = "varchar(20) check (share in ('공개','비공개')) default '비공개'")
    private String share;


    @ManyToOne(fetch = FetchType.LAZY)
    private User user;

    public void changePriority(Long priority){
        this.priority=priority;
    }
    public void  changeTitle(String title){
        this.title=title;
    }
    public void changeLocation(String location){
        this.location=location;
    }
    public void changeDescription(String description){
        this.description=description;
    }
    public void changeStart(LocalDate start){
        this.start= LocalDateTime.from(start);
    }
    public void changeEnd(LocalDate end){
        this.end= LocalDateTime.from(end);
    }
    public void changeShare(String share){
        this.share=share;
    }
    public void changeCategory(String category){
        this.category=category;
    }
}
```

DTO

PlanDTO.java

```java
package com.team.team_project.dto;


import lombok.*;

import java.time.LocalDateTime;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
@ToString
public class PlanDTO {
    private Long pno;
    private Long priority;
    private String title;
    private String location;
    private String description;
    private LocalDateTime start;
    private LocalDateTime end;
    private String category;
    private String share;
    private Long user_code;
}
```

Repository

PlanRepository.java(interface)

```java
package com.team.team_project.repository;

import com.team.team_project.entity.Plan;
import org.springframework.data.jpa.repository.JpaRepository;

public interface PlanRepository extends JpaRepository<Plan,Long> {
}
```

Service

PlanService.java(interface)

```java
package com.team.team_project.service;

import com.team.team_project.dto.PlanDTO;
import com.team.team_project.entity.Plan;
import com.team.team_project.entity.User;

public interface PlanService {

    default Plan dtoToEntity(PlanDTO dto){
        User user = User.builder()
                .code(dto.getUser_code())
                .build();
        Plan plan = Plan.builder()
                .pno(dto.getPno())
                .priority(dto.getPriority())
                .title(dto.getTitle())
                .category(dto.getCategory())
                .description(dto.getDescription())
                .share(dto.getShare())
                .location(dto.getLocation())
                .start(dto.getStart())
                .end(dto.getEnd())
                .user(user)
                .build();
        return plan;
    }

    default PlanDTO entityToDTO(Plan plan,User user){
        PlanDTO dto =PlanDTO.builder()
                .user_code(user.getCode())
                .pno(plan.getPno())
                .priority(plan.getPriority())
                .title(plan.getTitle())
                .description(plan.getDescription())
                .location(plan.getLocation())
                .category(plan.getCategory())
                .share(plan.getShare())
                .start(plan.getStart())
                .end(plan.getEnd())
                .build();
        return dto;
    }

     public Long makeAPlan(PlanDTO dto);
}
```

ServiceImpl

PlanServiceImpl.java

```java
package com.team.team_project.service;


import com.team.team_project.dto.PlanDTO;
import com.team.team_project.entity.Plan;
import com.team.team_project.repository.PlanRepository;
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.springframework.stereotype.Service;

@Service
@RequiredArgsConstructor

@Log4j2
public class PlanServiceImpl implements PlanService {
    private final PlanRepository planRepository;



    @Override
    public Long makeAPlan(PlanDTO dto) {
        Plan entity = dtoToEntity(dto);
        planRepository.save(entity);
        return entity.getPno();
    }
}
```

TEST

RepositoryTest.java

```java
@Autowired
    private PlanService planService;

//    @Test
    public void makeaplan(){
        PlanDTO dto =PlanDTO.builder()
                .user_code(3L)
                .pno(1L)
                .priority(10L)
                .title("전기뱀장어")
                .description("이해하자")
                .location("장충동")
                .category("족발")
                .share("공개")
                .start(LocalDateTime.parse("2002-01-23T09:30:00"))
                .end(LocalDateTime.parse("2002-01-23T09:30:00"))
                .build();
        System.out.println(planService.makeAPlan(dto));


    }

    @Test
    public void hundredmakeplan(){
        IntStream.rangeClosed(1,100).forEach(i->{
            PlanDTO dto =PlanDTO.builder()
                    .user_code((long)i)
                    .title("테스트"+i)
                    .description("TEST_1"+i)
                    .location("TEST"+i+"번지")
                    .start(LocalDateTime.parse("2002-01-23T09:30:00"))
                    .end(LocalDateTime.parse("2002-01-23T09:30:00"))
                    .build();
            System.out.println(planService.makeAPlan(dto));
        });
    }
}
```

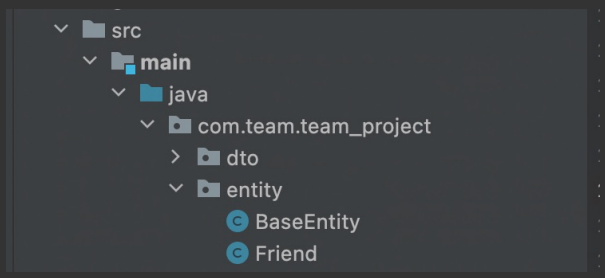
MySQL_Properties

```
CREATE TABLE `plan` (
  `pno` bigint NOT NULL AUTO_INCREMENT,
  `moddate` datetime(6) DEFAULT NULL,
  `regdate` datetime(6) DEFAULT NULL,
  `category` varchar(20) DEFAULT 'No Category',
  `description` varchar(255) DEFAULT NULL,
  `end` datetime(6) NOT NULL,
  `location` varchar(255) DEFAULT NULL,
  `priority` bigint DEFAULT NULL,
  `share` varchar(20) DEFAULT '비공개',
  `start` datetime(6) NOT NULL,
  `title` varchar(100) NOT NULL,
  `user_code` bigint DEFAULT NULL,
  PRIMARY KEY (`pno`),
  KEY `FKm46h77ilxf1ji9k1qs22fgavl` (`user_code`),
  CONSTRAINT `FKm46h77ilxf1ji9k1qs22fgavl` FOREIGN KEY (`user_code`)
REFERENCES `user` (`code`),
  CONSTRAINT `plan_chk_1` CHECK ((`share` in (_utf8mb4'공개',_utf8mb4'비공개')))
) ENGINE=InnoDB AUTO_INCREMENT=201 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci
```

Friends 테이블 생성 및 연결 하기

```sql
# 상태코드 대기, 수락, 거절, 취소
create table Friend(
    request int,
    response int,
    status varchar(5),
    date DATE,
    constraint User_FK_Friend_request foreign key (request) references user(code) on delete cascade,
    constraint User_FK_Friend_response foreign key (response) references user(code) on delete cascade
) DEFAULT CHARSET=utf8;
```

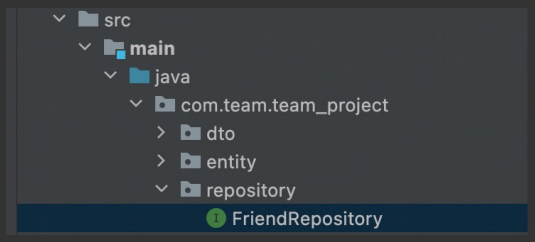
Friend.java(Entity)



Friend.java

```java
package com.team.team_project.entity;

import lombok.*;
import org.hibernate.annotations.DynamicInsert;

import javax.persistence.*;
@Entity
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Getter
@ToString(exclude = {"request","response"})
@DynamicInsert
public class Friend extends BaseEntity{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long fno;

    @Column(columnDefinition = "varchar(5) check (status in ('수락','거절','차단')) default '요청중'")
    private String status;

    @ManyToOne(fetch = FetchType.LAZY)
    private User request;

    @ManyToOne(fetch = FetchType.LAZY)
    private User response;

    public void changeStatus(String status){
        this.status=status;
    }
}
```

FriendRepository(interface) 생성



FriendRepository.java

```java
package com.team.team_project.repository;

import com.team.team_project.entity.Friend;
import org.springframework.data.jpa.repository.JpaRepository;

public interface FriendRepository extends JpaRepository<Friend,Long> {
}
```

TEST

MySQL

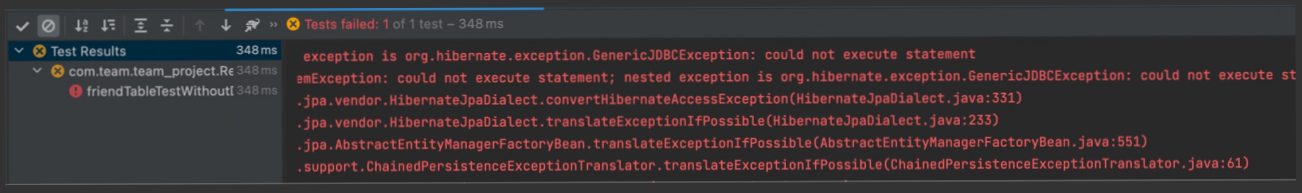
```sql
CREATE TABLE `friend` (
  `fno` bigint NOT NULL AUTO_INCREMENT,
  `moddate` datetime(6) DEFAULT NULL,
  `regdate` datetime(6) DEFAULT NULL,
  `status` varchar(5) DEFAULT '요청중',
  `request_code` bigint DEFAULT NULL,
  `response_code` bigint DEFAULT NULL,
  PRIMARY KEY (`fno`),
  KEY `FK62ceykl58m6bo87iwf0thk3lu` (`request_code`),
  KEY `FKnoxw6sxhu1yy7loknnyye0khh` (`response_code`),
  CONSTRAINT `FK62ceykl58m6bo87iwf0thk3lu` FOREIGN KEY (`request_code`) REFERENCES `user` (`code`),
  CONSTRAINT `FKnoxw6sxhu1yy7loknnyye0khh` FOREIGN KEY (`response_code`) REFERENCES `user` (`code`),
  CONSTRAINT `friend_chk_1` CHECK ((`status` in (_utf8mb4'수락',_utf8mb4'거절',_utf8mb4'차단')))
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

TEST ———

RepositoryTest.java

```java
@Autowired
    private FriendRepository friendRepository;
    @Test
    public void friendTableTestWithoutDto(){
        User user1 = User.builder()
                .code(1L)
                .build();
        User user2 = User.builder()
                .code(2L)
                .build();
        Friend friend = Friend.builder()
                .response(user1)
                .request(user2)
                .build();
        friendRepository.save(friend);
    }
}
```

—— RESULT



MySQL

```sql
CREATE TABLE `friend` (
  `fno` bigint NOT NULL AUTO_INCREMENT,
  `moddate` datetime(6) DEFAULT NULL,
  `regdate` datetime(6) DEFAULT NULL,
  `status` varchar(5) DEFAULT '요청중',
  `request_code` bigint DEFAULT NULL,
  `response_code` bigint DEFAULT NULL,
  PRIMARY KEY (`fno`),
  KEY `FK62ceykl58m6bo87iwf0thk3lu` (`request_code`),
  KEY `FKnoxw6sxhu1yy7loknnyye0khh` (`response_code`),
  CONSTRAINT `FK62ceykl58m6bo87iwf0thk3lu` FOREIGN KEY (`request_code`) REFERENCES `user` (`code`),
  CONSTRAINT `FKnoxw6sxhu1yy7loknnyye0khh` FOREIGN KEY (`response_code`) REFERENCES `user` (`code`),
  CONSTRAINT `friend_chk_1` CHECK ((`status` in (_utf8mb4'수락',_utf8mb4'거절',_utf8mb4'차단')))
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

——〉 DB 생성은 어째 잘 되는데 안들어간다.

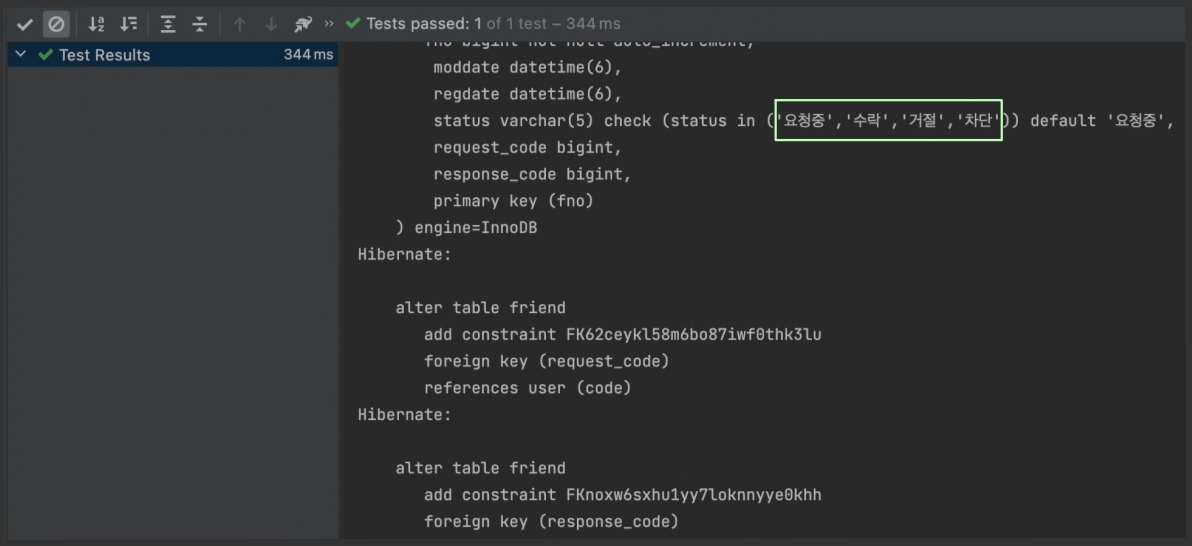
예상 원인 : Default 로 '요청중' 을 줬는데 Check 항목에 포함이 되지 않음

Entity 수정

Friend.java

```java
package com.team.team_project.entity;

import lombok.*;
import org.hibernate.annotations.DynamicInsert;

import javax.persistence.*;
@Entity
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Getter
@ToString(exclude = {"request","response"})
@DynamicInsert
public class Friend extends BaseEntity{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long fno;

    @Column(columnDefinition = "varchar(5) check (status in ('요청중','수락','거절','차단')) default '요청중'")
    private String status;

    @ManyToOne(fetch = FetchType.LAZY)
    private User request;

    @ManyToOne(fetch = FetchType.LAZY)
    private User response;

    public void changeStatus(String status){
        this.status=status;
    }
}
```

TEST RESULT



MYSQL.FRIEND_PROPERTIES

```
CREATE TABLE `friend` (
  `fno` bigint NOT NULL AUTO_INCREMENT,
  `moddate` datetime(6) DEFAULT NULL,
  `regdate` datetime(6) DEFAULT NULL,
  `status` varchar(5) DEFAULT '요청중',
  `request_code` bigint DEFAULT NULL,
  `response_code` bigint DEFAULT NULL,
  PRIMARY KEY (`fno`),
  KEY `FK62ceykl58m6bo87iwf0thk3lu` (`request_code`),
  KEY `FKnoxw6sxhu1yy7loknnyye0khh` (`response_code`),
  CONSTRAINT `FK62ceykl58m6bo87iwf0thk3lu` FOREIGN KEY (`request_code`) REFERENCES `user` (`code`),
  CONSTRAINT `FKnoxw6sxhu1yy7loknnyye0khh` FOREIGN KEY (`response_code`) REFERENCES `user` (`code`),
  CONSTRAINT `friend_chk_1` CHECK ((`status` in (_utf8mb4'요청중',_utf8mb4'수락',_utf8mb4'거절',_utf8mb4'차단')))
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```