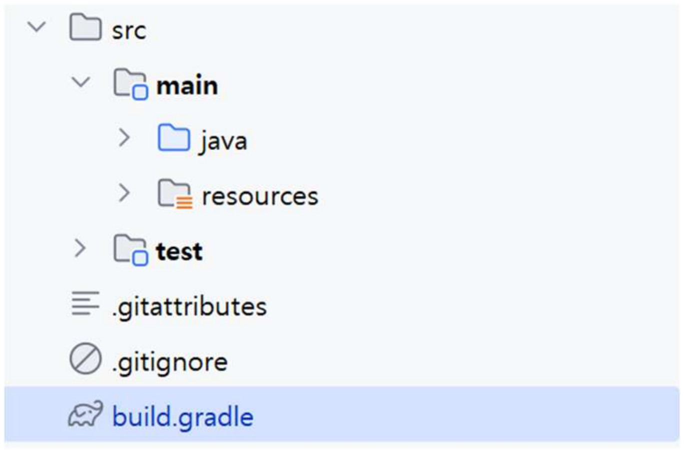
서비스 로직 작성

User 서비스 회원가입

User 프로젝트 종속성 추가



```
src
  main
    java
    resources
  test
.gitattributes
.gitignore
build.gradle
```

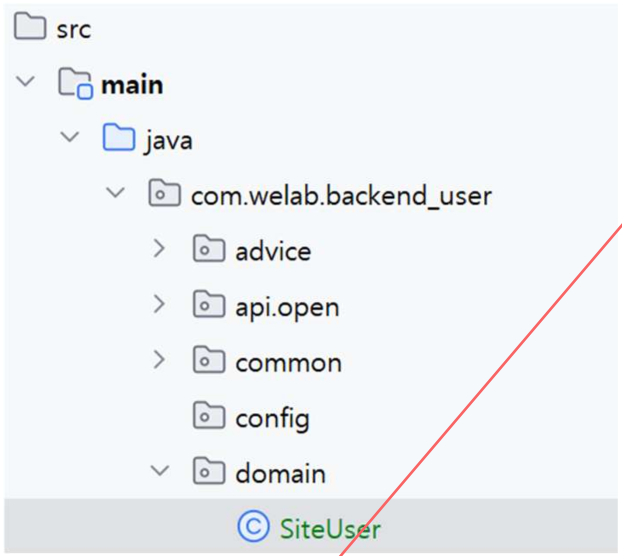
```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.cloud:spring-cloud-starter-loadbalancer'
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'
    implementation 'org.springframework.cloud:spring-cloud-starter-openfeign'
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    runtimeOnly 'com.mysql:mysql-connector-j:8.4.0'
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'

    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
}
```

User 프로젝트 설정

```yaml
spring:
  datasource:
    url: jdbc:mysql://localhost:13306/user?serverTimezone=UTC&useSSL=true&autoReconnect=true&useUnicode=true&characterEncoding=utf-8
    username: user
    password: 1234
    driver-class-name: com.mysql.cj.jdbc.Driver
    hikari:
      connection-test-query: SELECT 1  # HikariCP 유효성 검사 추가
          validation-timeout: 5000
  jpa:
    hibernate:
      ddl-auto: create # 오직 테스트 환경에서만
        generate-ddl: true # 오직 테스트 환경에서만
        show-sql: true
    open-in-view: false
#    properties:
#      hibernate:
#        dialect: org.hibernate.dialect.MySQL8Dialect
```

application-local.yml

User 프로젝트 코드 작성 – SiteUser Entity

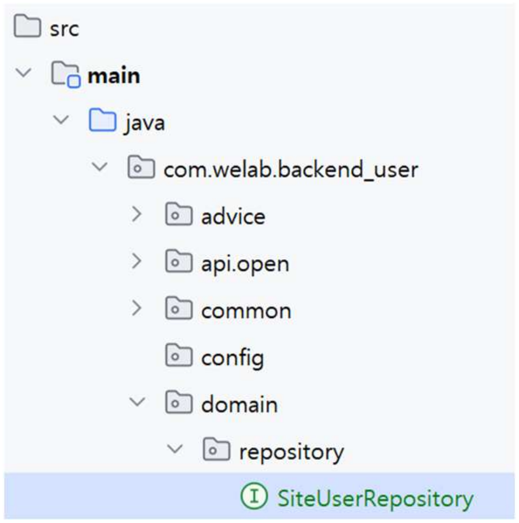


```
@Slf4j
@Entity
@Table(name = "site_user")
public class SiteUser {
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Getter
    @Setter
    private Long id;

    @Column(name = "user_id", unique = true, nullable = false)
    @Getter @Setter
    private String userId;

    @Column(name = "password", nullable = false)
    @Getter @Setter
    private String password;

    @Column(name = "phone_number", nullable = false)
    @Getter @Setter
    private String phoneNumber;

    @Column(name = "deleted", nullable = false)
    @Getter @Setter
    private Boolean deleted = false;
}
```

NullException 과 같은 문제를 피하기 위해
개별로 @Getter @Setter 적용
nullable = true 시,
아래의 코드를 통해 null 값 방지 가능

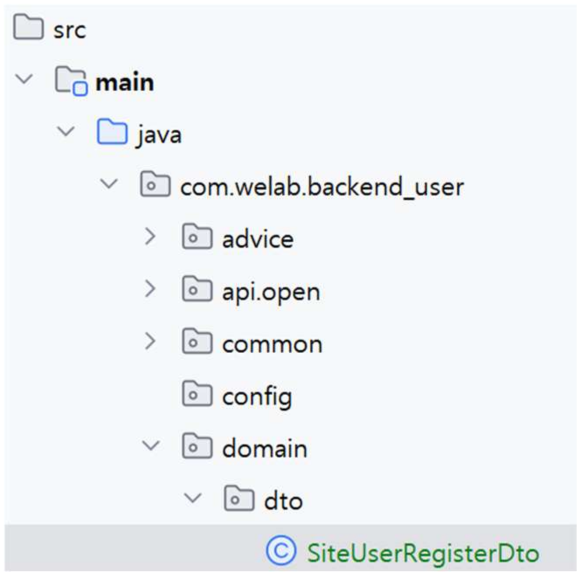
```
public Optional<String>
getPhoneNumber() {
   return Optional.of(this.phoneNumber);
}
```

User 프로젝트 코드 작성 – SiteUserRepository


src
main
java
com.welab.backend_user
advice
api.open
common
config
domain
repository
SiteUserRepository

```
@Repository
public interface SiteUserRepository extends JpaRepository<SiteUser, Long> {
    SiteUser findByUserId(String userId);
}
```

User 프로젝트 코드 작성 – SiteUserRegisterDto



```java
@Getter
@Setter
public class SiteUserRegisterDto {
    private String userId;
    private String password;
    private String phoneNumber;
}
```

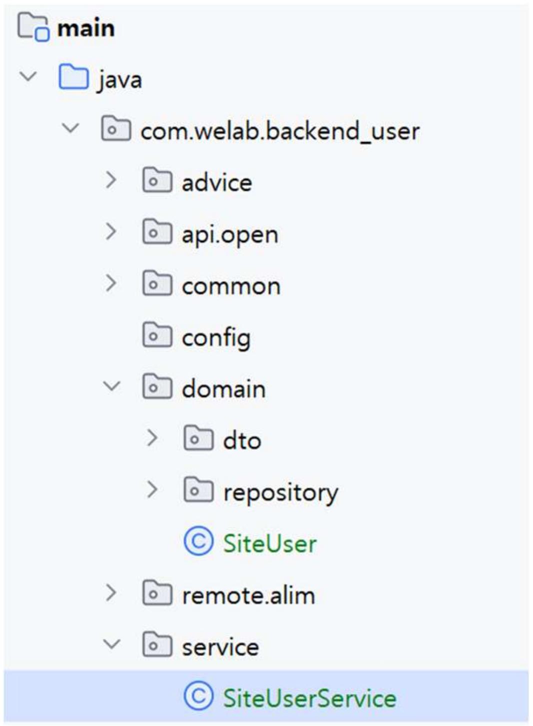
SiteUserRegisterDto constraints 적용

```java
@Getter
@Setter
public class SiteUserRegisterDto {
    @NotBlank(message = "아이디를 입력하세요.")
    private String userId;

    @NotBlank(message = "비밀번호를 입력하세요.")
    private String password;

    @NotBlank(message = "전화번호를 입력하세요.")
    private String phoneNumber;
}
```

SiteUserRegisterDto -> SiteUser

```java
@Getter
@Setter
public class SiteUserRegisterDto {
    @NotBlank(message = "아이디를 입력하세요.")
    private String userId;

    @NotBlank(message = "비밀번호를 입력하세요.")
    private String password;

    @NotBlank(message = "전화번호를 입력하세요.")
    private String phoneNumber;

    public SiteUser toEntity() {
        SiteUser siteUser = new SiteUser();

        siteUser.setUserId(this.userId);
        siteUser.setPhoneNumber(this.phoneNumber);
        // TODO: SHA1 또는 SHA256으로 password를 해시 값으로 변환
         String hashedPassword = this.password;
        siteUser.setPassword(hashedPassword);

        return siteUser;
    }
}
```

User 프로젝트 코드 작성 – SiteUserService



```
@Slf4j
@Service
@RequiredArgsConstructor
public class SiteUserService {
    private final SiteUserRepository siteUserRepository;
}
```

SiteUserService – 회원가입

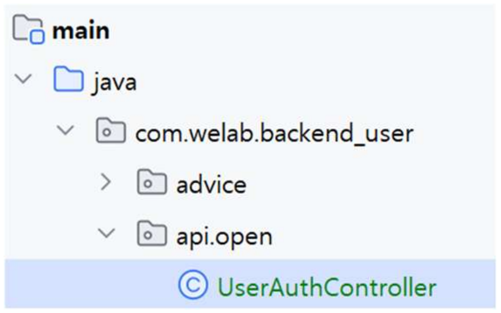
```java
@Slf4j
@Service
@RequiredArgsConstructor
public class SiteUserService {
    private final SiteUserRepository siteUserRepository;

    @Transactional
    public void registerUser(SiteUserRegisterDto registerDto) {
        SiteUser siteUser = registerDto.toEntity();

        siteUserRepository.save(siteUser);
    }
}
```

[다음과 같은 DI 방법 사용 이유]
1. 순환참조 방지
2. final을 통한 변수 변경 불가

Entity와 Dto 사용 일관성
DB에 접근 시, Entity 사용

```java
// Transactional 적용 안하고 register() 메소드 실행 방법
public void registerAct(SiteUserRegisterDto registerDto) {
    this.register(registerDto);
}
```

User 프로젝트 코드 작성 – UserAuthController



```
@Slf4j
@RestController
@RequestMapping(value = "/api/user/v1/auth", produces = MediaType.APPLICATION_JSON_VALUE)
@RequiredArgsConstructor
public class UserAuthController {
    private final SiteUserService siteUserService;

    @PostMapping(value = "/register")
    public ApiResponseDto<String> register(@RequestBody @Valid SiteUserRegisterDto registerDto) {
        siteUserService.registerUser(registerDto);
        return ApiResponseDto.defaultOk();
    }
}
```

SiteUserRegisterDto의
constraints 검증
(ex. @NotBlank)