



E-commerce - User Service

🕒 작성 일시	@2022년 11월 11일 오후 11:35
🕒 최종 편집 일시	@2022년 11월 22일 오후 11:59
📌 유형	Spring Cloud로 개발하는 마이크로서비스 애플리케이션(MSA)
👤 작성자	
👥 참석자	

[목표](#)

[API](#)

[프로젝트 생성](#)

[application.xxx](#) 에서 변수로 할당하는 방법

[PostgreSQL 연동](#)

[JPA 설정](#)

[Security 연동](#)

[User Microservice 사용자 조회 - git 참고](#)

목표

- 신규 회원 등록
- 회원 로그인
- 회원 정보 확인
- 회원 정보 수정
- 상품 주문
- 주문 내역 확인

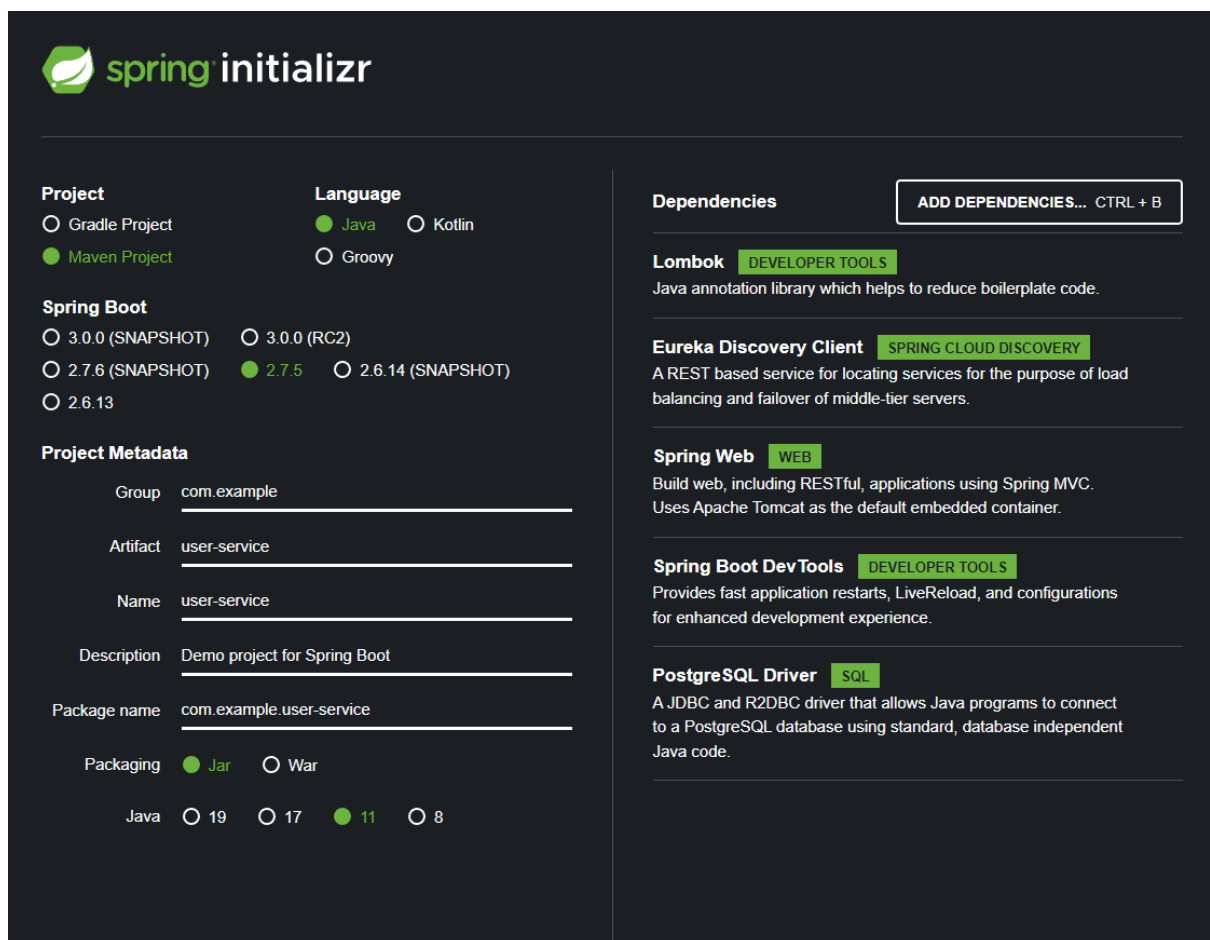
API

기능	URL (API Gateway)	URL (API Gateway 미사용)	HTTP Method
사용자 정보 등록	/user-service/users	/users	POST

기능	URL (API Gateway)	URL (API Gateway 미사용)	HTTP Method
전체 사용자 조회	/user-service/users	/users	GET
사용자 정보, 주문 내역 조회	/user-service/users/{user_id}	/users/{user_id}	GET
작동 상태 확인	/user-service/user/health_check	/user/health_check	GET
환영 메시지	/user-service/user/welcome	/user/welcome	GET

프로젝트 생성

(강의는 H2 를 사용했지만 나는 postgresql 함으로 dependency 에 postgresql driver 를 추가한다.)



The image shows the Spring Initializr web interface for configuring a new project. The interface is divided into several sections:

- Project:**
 - ☐ Gradle Project
 - ☒ Maven Project
- Language:**
 - ☒ Java
 - ☐ Kotlin
 - ☐ Groovy
- Spring Boot:**
 - ☐ 3.0.0 (SNAPSHOT)
 - ☐ 3.0.0 (RC2)
 - ☐ 2.7.6 (SNAPSHOT)
 - ☒ 2.7.5
 - ☐ 2.6.14 (SNAPSHOT)
 - ☐ 2.6.13
- Project Metadata:**
 - Group:
 - Artifact:
 - Name:
 - Description:
 - Package name:
 - Packaging: ☒ Jar ☐ War
 - Java: ☐ 19 ☐ 17 ☒ 11 ☐ 8
- Dependencies:**
 - Lombok** (DEVELOPER TOOLS): Java annotation library which helps to reduce boilerplate code.
 - Eureka Discovery Client** (SPRING CLOUD DISCOVERY): A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.
 - Spring Web** (WEB): Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
 - Spring Boot DevTools** (DEVELOPER TOOLS): Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
 - PostgreSQL Driver** (SQL): A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.

application.xxx 에서 변수로 할당하는 방법

- Environment 를 할당 받고 getProperty("aaa.bbb") 메소드를 사용
- @Value(\${aaa.bbb})

PostgreSQL 연동

- application.yml

```
#####  
#           DB connection           #  
#####  
datasource:  
  driver-class-name: org.postgresql.Driver  
  url: jdbc:postgresql://localhost:5432/ecc  
  username: ecc  
  password: ecc1234!
```

JPA 설정

- validation Check 를 하기 위해 새 dependency를 추가.
 - @NotNull, @Size, @Email 등

```
<dependency>  
  <groupId>jakarta.validation</groupId>  
  <artifactId>jakarta.validation-api</artifactId>  
</dependency>
```

- jpa 를 사용하기 위해 새 dependency 추가.

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

- 하나의 객체를 다른 객체로 변환 시켜주는 ModelMapper 를 사용함으로 새 dependency 추가. 및 예제

```
<dependency>
  <groupId>org.modelmapper</groupId>
  <artifactId>modelmapper</artifactId>
  <version>2.3.8</version>
</dependency>
```

```
@Override
public UserDto createUser(UserDto userDto) {
    userDto.setUserId(UUID.randomUUID().toString());

    ModelMapper mapper = new ModelMapper();
    // 매핑 할 수 있도록 setting config
    mapper.getConfiguration().setMatchingStrategy(MatchingStrategies.STRICT);
    // 해당 객체를 해당 class 객체로 매핑.
    UserEntity userEntity = mapper.map(userDto, UserEntity.class);

    userEntity.setEncryptedPwd("encrypted_password");

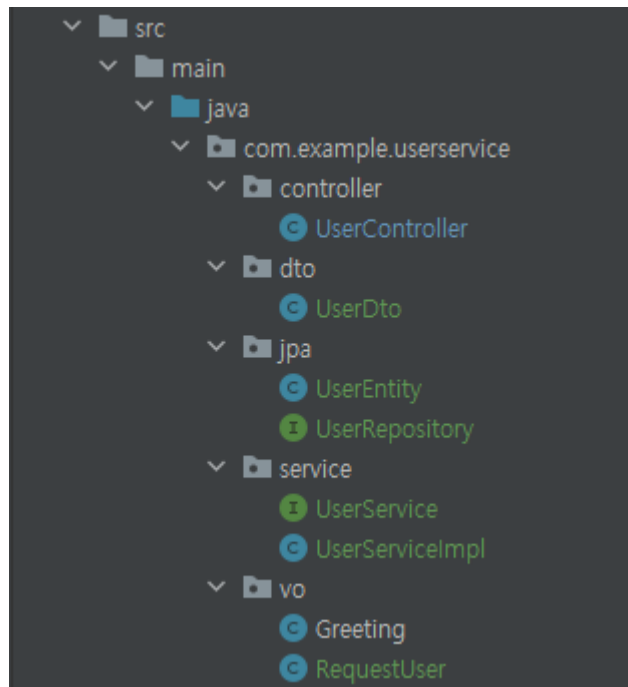
    userRepository.save(userEntity);

    return null;
}
```

- Postgresql 로 jpa 사용하기 위해 yml 추가 작성

```
spring:
  application:
    name: user-service
#####
#      DB connection      #
#####
datasource:
  driver-class-name: org.postgresql.Driver
  url: jdbc:postgresql://localhost:5432/ecc
  username: ecc
  password: ecc1234!
jpa:
  hibernate:
    dialect: org.hibernate.dialect.PostgreSQL10Dialect
    ddl-auto: update
  properties:
    hibernate:
      format_sql: true
    show-sql: true
```

- 아래 내용처럼 패키지를 구성한다. (상세 내용은 소스코드 참고)
- controller(RequestUser) → service(UserDto) → Repository[CrudRepository] (UserEntity)



Security 연동

- Authentication + Authorization
 - dependency 추가

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

- Security Configuration 클래스 + @EnableWebSecurity 추가

```
@Configuration
@EnableWebSecurity
public class WebSecurity extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        // https://velog.io/@woohobi/Spring-security-csrf%EB%9E%80
        // csrf 사용 안함.
        http.csrf().disable();
        http.authorizeRequests().antMatchers("/users/**").permitAll();
    }

    @Bean
    public BCryptPasswordEncoder passwordEncoder() {
```

```

        return new BCryptPasswordEncoder();
    }
}

```

- Authentication → configure 메서드 재 정의
- Password encode를 위한 BCryptPasswordEncoder 빈 정의
- Authorization → configure(HttpSecurity http) 메서드 재 정의

User Microservice 와 Spring Cloud Gateway 연동

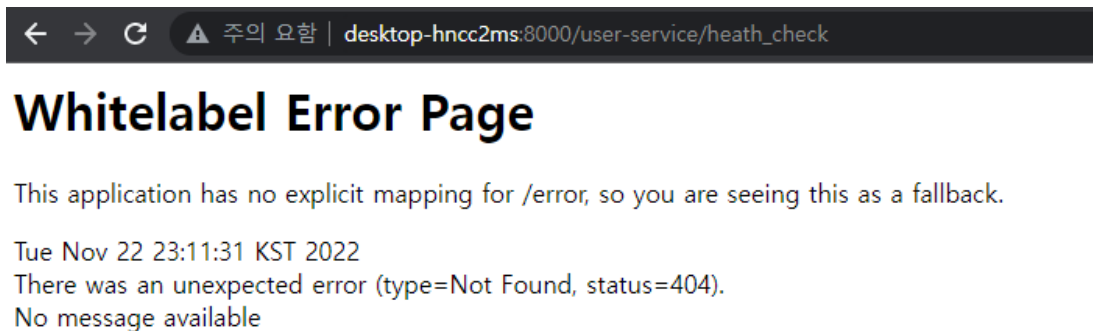
- API Gateway Service 의 yml 에 해당 내용 추가

```

routes:
- id: user-service
  uri: lb://USER-SERVICE
  predicates:
  - Path=/user-service/**

```

- API Gateway에서 user-service 호출시에 정상적으로 나와야 할 것 같지만 404가 뜬다.



- User Service 의 URI 와 API Gateway URI 가 다름.
- user-service 라는 전치사가 붙기 때문에, User Service 에서 인식하지 못하고 404를 뱉는다.
- API Gateway 에 Service를 등록시 해당 Service의 전치사를 똑같이 추가하도록 한다.

```

@RestController
@RequestMapping("/user-service")

```

```
public class UserController {
```

User Microservice 사용자 조회 - git 참고

- UserDto getUserById(String userId)
- List<UserDto> getUserByAll()

