Address

Application 설정 및 설명.

1.[**빌드**](#_eyhyo45kpx9u) **2**

**2.**[**동작**](#_9esynv3icjv9) **4**

[2.1 CONTROLLER](#_786r2z2gwidr) 4

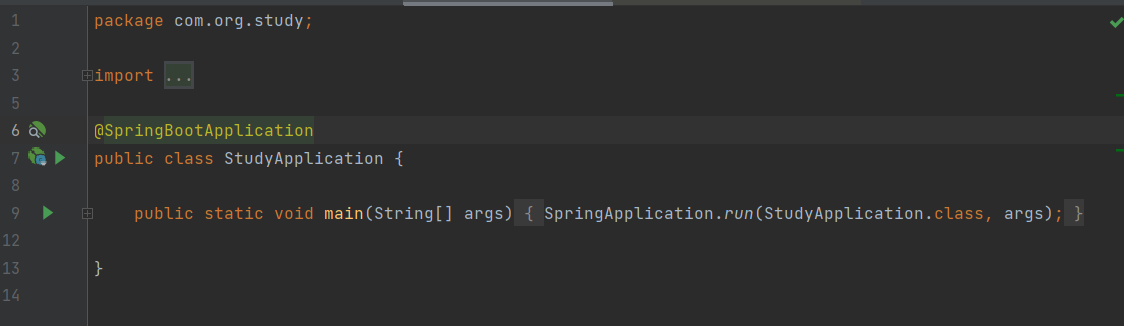
[2.2 SERVICE](#_kf5ywds55ryy) 6

[2.3 MAPPER ( DAO )](#_swovvfm24dmq) 7

### 빌드

APPLICATION의 메인에서 실행되는 spring application 은 기본 설정(@org.springframework.boot.autoconfigure.EnableAutoConfiguration

)을 따라서 실행



SpringBootApplication 어노테이션을 통해 내부에서는 / 타겟 / 적용 범위 / 기본 설정 값 / 컴포넌트 스캔등을 통해 컨트롤러와 서비스와 같은 스프링에서 제공하는 기능등을

리소스에 등록.

빌드 시,

web.xml 파일에 의한 설정이 아닌

application.properties으로 구현.

application.properties :

# jdbc 설정 MySQL과 연동하고, 연동 시 사용되는 쿼리를 로그로 기록하기 위한 설정.

spring.datasource.driver-class-name=net.sf.log4jdbc.sql.jdbcapi.DriverSpy

spring.datasource.url=jdbc:log4jdbc:mysql://localhost:3306/study?autoReconnect=true

spring.datasource.username=root

spring.datasource.password=1234

# MyBatis SQL 파일의 경로를 지정하기위한 설정 :

spring.mvc.pathmatch.matching-strategy=ant\_path\_matcher

management.info.env.enabled=true

mybatis.mapper-locations=classpath:mapper/\*\*/\*\_SQL.xml

applicaiton.properties 이후 ServletInitializer 동작.

DB 와의 커넥션을 위해 SqlSessionFactory 를 구현해서 bean에 등록.

| @Bean(name = "sqlSessionFactory")  public SqlSessionFactory sqlSessionFactory( @Qualifier("dataSource")DataSource dataSource, ApplicationContext applicationContext) throws Exception {  SqlSessionFactoryBean sqlSessionFactory = new SqlSessionFactoryBean();  sqlSessionFactory.setDataSource(dataSource);  sqlSessionFactory.setMapperLocations(new PathMatchingResourcePatternResolver().getResource("mapper/\*\*/\*.xml"));  return sqlSessionFactory.getObject();  } |
| --- |

등록 한 “sqlSessionFactory” 빈을 sqlSessionTemplate으로 설정.

### 동작

사용자가 HTTP 방식을 통해 서버의 api를 호출한다.

Restful하게 구현을 하기 위해

제공하는 API는 리소스를 구분해서 명시하고, HTTP Method를 나눠서 설정

RestController로 구현하고, 요청과 응답은 JSON 형식 통신이며, HttpStatus를 같이 리턴한다.

#### 

#### 2.1 CONTROLLER

GET / POST / PUT / PATCH / DELETE 메소드를 통해 구현.

|  |
| --- |

사용자가 잘못된 파라미터로 요청을 하거나, service / dao 단에서 일어나는 에러를 캐치해서 return하거나 재처리하는 요청의 최앞단.

사용자의 요청에 대한 최종 결과값은 Service를 호출하여 정보를 가공 / 수집해서 가져온다.

Autowired 혹은 bean에 미리 등록한 서비스 정보를 호출하여 실제 비즈니스 로직을 수행 하게 함.

|  |
| --- |

|  |
| --- |

#### 2.2 SERVICE

인터페이스를 사용해 추상적으로 서비스 메서드를 구현.

|  |
| --- |

인터페이스를 구현한 ServiceImpl 클래스로 Controller에서 받은 요청값에 대한 실제 비즈니스 로직을 구현한다.

|  |
| --- |

비즈니스 로직을 통해 가공한 데이터 혹은 새로 생성한 데이터들을 MAPPER를 통해 DB와 통신하도록 구현.

#### 2.3 MAPPER ( DAO )

SQLTEMPLATE을 호출. 이 APP에서는 JPA가 아닌 MyBatis로 구현.

서비스에서 가공 / 사용자가 요청한 정보를 통해 DB의 정보를 수집.

|  |
| --- |

DB와 통신하여 주고받는 데이터 값은 SQL 문과 ResultSetTable을 log4j 설정을 통해 서버로그 또는 파일로 기록.

|  |
| --- |

요청자는 다음과 같은 형식으로 요청한 데이터, HttpStatus를 리턴받는다.

|  |
| --- |