Spring Data JPA, Spring Boot, Querydsl실습, Query Method, @Query, @NamedQuery, 페이징, 서브쿼리, 조인)

■ 지금까지 학습한 Spring Boot, Spring Data JPA, Spring WEB MVC, Querydsl @NamedQuery, @Query, 메소드 이름으로 쿼리생성(Query Method), 페이징처리, 서브쿼리, 조인 기본 예제를 마리아DB를 이용하여 작성해 보자.

STS -> Spring Starter Project
project name : jpawebexam2
Type : MAVEN
package : jpa
Core : Lombok
SQL -> JPA, MySQL
Web -> Web 선택

Querydsl MAVEN 설정은 아래 URL에서 참조
http://ojc.asia/bbs/board.php?bo_table=LecSpring&wr_id=543

마리아 DB 설치는 다음 URL 참조
http://ojc.asia/bbs/board.php?bo_table=LecSpring&wr_id=524

롬복(Lombok)설치는 다음 URL 참조
http://ojc.asia/bbs/board.php?bo_table=LecSpring&wr_id=561

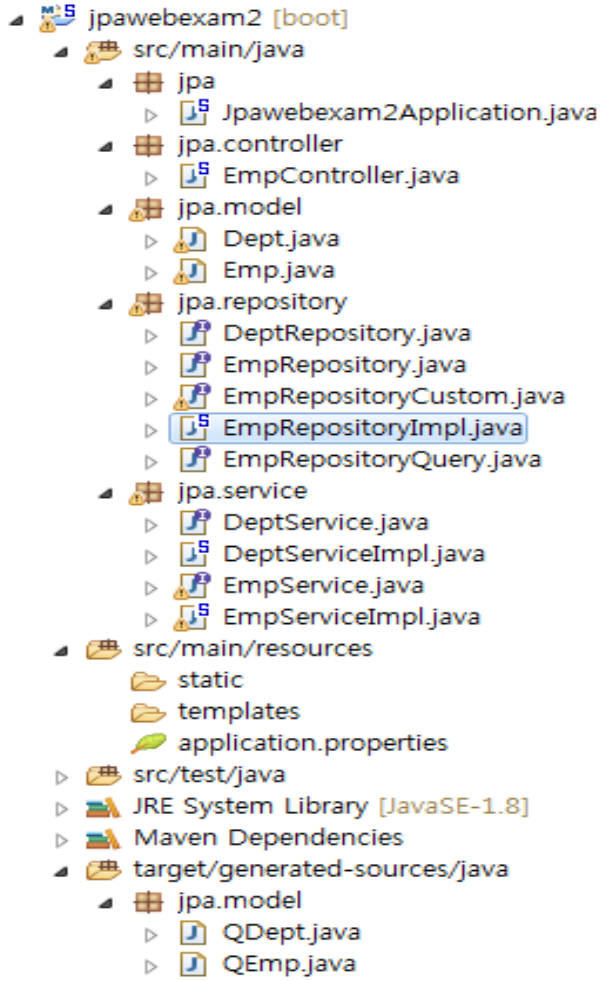
[pom.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>

	<groupId>ojc.asia</groupId>
	<artifactId>querydsl</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>jar</packaging>
```

```
<name>jpawebexam2</name>
<description>Spring Data JPA, Querydsl Example</description>

<parent>
	<groupId>org.springframework.boot</groupId>
	<artifactId>spring-boot-starter-parent</artifactId>
	<version>1.3.1.RELEASE</version>
	<relativePath /> <!-- lookup parent from repository -->
</parent>

<properties>
	<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
	<java.version>1.8</java.version>
	<querydsl.version>4.0.8</querydsl.version>
</properties>

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

	<dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
		<scope>runtime</scope>
	</dependency>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-test</artifactId>
		<scope>test</scope>
	</dependency>

	<dependency>
```

```
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
                <version>1.16.6</version>
            </dependency>

            <dependency>
                <groupId>com.querydsl</groupId>
                <artifactId>querydsl-apt</artifactId>
                <version>${querydsl.version}</version>
                <scope>provided</scope>
            </dependency>

            <dependency>
                <groupId>com.querydsl</groupId>
                <artifactId>querydsl-jpa</artifactId>
                <version>${querydsl.version}</version>
            </dependency>

            <dependency>
                <groupId>org.osgi</groupId>
                <artifactId>org.osgi.compendium</artifactId>
                <version>5.0.0</version>
            </dependency>
        </dependencies>

        <build>
            <plugins>
                <plugin>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-maven-plugin</artifactId>
                </plugin>

                <plugin>
                    <groupId>com.mysema.maven</groupId>
                    <artifactId>apt-maven-plugin</artifactId>
                    <version>1.1.3</version>
                    <executions>
                        <execution>
                            <goals>
```

```
                                                    <goal>process</goal>
                                                </goals>
                                                <configuration>
                                                    <outputDirectory>target/generated-
sources/java</outputDirectory>

            <processor>com.querydsl.apt.jpa.JPAAnnotationProcessor</processor>
                                                </configuration>
                                            </execution>
                                        </executions>
                                    </plugin>
                                </plugins>
            </build>
</project>
```



[Jpawebexam2Application.java]

```
package jpa;
import org.springframework.boot.SpringApplication;import
org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class Jpawebexam2Application {
public static void main(String[] args) {
        SpringApplication.run(Jpawebexam1Application.class, args);
    }
}
```

[application.properties]

```
spring.datasource.platform=mysql
spring.datasource.sql-script-encoding=UTF-8
spring.datasource.url=jdbc:mysql://localhost/jpawebexam2?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=1111
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
logging.level.jpa=DEBUG
```

[Emp.java]

```
package jpa.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQuery;

import lombok.Getter;
import lombok.Setter;

@Entity
@Getter
@Setter
@NamedQuery(name="Emp.findBySalNamed",
            query="select e from Emp e where e.sal > :sal")
```

```
public class Emp {
	@Id
	@GeneratedValue
	private Long empno;
	private String ename;
	private String job;
	private Long sal;

	@ManyToOne
	@JoinColumn(name = "deptno")
	private Dept dept;
}
```

[Dept.java]

```
package jpa.model;

import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToMany;

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Entity
@Getter
@Setter
@NoArgsConstructor
@RequiredArgsConstructor
public class Dept   {
	@Id
	@GeneratedValue
	private Long deptno;
```

```
	@Column(unique=true)
	@NonNull
	private String dname;
}
```

[DeptRepository.java]

```
package jpa.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import jpa.model.Dept;

public interface DeptRepository extends JpaRepository<Dept, Long>{
	Dept findByDname(String dname); //query method
}
```

[EmpRepository.java] - 레포지토리 클래스

```
package jpa.repository;

import java.util.List;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;

import jpa.model.Emp;

/*
 * 기본 JpaRepository외 사용자정의 인터페이스 EmpRepositoryCustom을
 * 상속받고 Query Method 2개를 정의하고 있다.
 */
public interface EmpRepository extends JpaRepository<Emp, Long>, EmpRepositoryCustom{
	// sal값이 지정된값보다 크거나 같은조건으로, 페이징 기능을 이용해 추출
	// 메소드 명으로 쿼리 자동생성
	Page<Emp> findBySalGreaterThan(Long sal, Pageable pageable);

	// 급여로 조회하는데 이름내림차순으로 처음3건만 추출
```

```
        // 메소드 명으로 쿼리 자동생성
        List<Emp> findFirst3BySalBetweenOrderByEnameDesc(Long sal1, Long Sal2);
}
```

[EmpRepositoryCustom.java] – 사용자 정의 레포지토리

```
package jpa.repository;

import java.util.List;
import com.querydsl.core.Tuple;
import jpa.model.Dept;
import jpa.model.Emp;

/*
 * 사용자 정의 인터페이스, 기본으로 제공되는 JpaRepository이외의
 * 사용자 쿼리 작성할 때 만드는 인퍼테이스이며 Query Mathod 및
 * NamedQuery 처리를 위한 메소드를 정의하고 있다.
 */
public interface EmpRepositoryCustom {

        //NamedQuery 처리용
        List<Emp> findBySalNamed(Long sal);

        //Querydsl 처리용
        List<Emp> selectByJobOrderByEnameDesc(String job);
        Long deleteByJob(String job);
        Long updateByEmpno(Long empno, String newEname);
        List<Tuple> selectEnameJobByEmpno(Long empno);
        List<Tuple> selectEmpEnameDnameJoinDept(Long deptno);   //Join
        List<Emp> selectEmpMaxSal();              //subquery
        List<Emp> selectEmpMaxSalOfDept();     //subquery
        List<Emp> selectEmpGreaterThanAvgSal();   //subquery
        List<Emp> selectEmpEqualsEmpno(Long empno);   //subquery
        List<Emp> selectEmpMaxSalTop3();   //subquery
        List<String> selectDeptExistsEmp();   //subquery
}
```

[EmpRepositoryImpl.java]

```
package jpa.repository;
```

```
import static jpa.model.QDept.dept;
import static jpa.model.QEmp.emp;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;
import com.querydsl.core.Tuple;
import com.querydsl.jpa.JPAExpressions;
import com.querydsl.jpa.impl.JPADeleteClause;
import com.querydsl.jpa.impl.JPAQuery;
import com.querydsl.jpa.impl.JPAUpdateClause;
import jpa.model.Emp;
import jpa.model.QEmp;

// 사용자정의인터페이스 구현
// Querydsl용 인터페이스 구현
@Repository
public class EmpRepositoryImpl implements EmpRepositoryCustom {

	@PersistenceContext
	EntityManager em;

	////////////////////////////////// NamedQuery 처리 메소드
	// Emp엔티티에서 정의된 NamedQuery 사용
	public List<Emp> findBySalNamed(Long sal) {
		List<Emp> result = em.createNamedQuery("Emp.findBySalNamed", Emp.class)
		.setParameter("sal", 2000L)
		.getResultList();

		return result;
	}

	////////////////////////////////// Querydsl용 메소드
	@Override
	/* Emp 테이블에서 job을 조건으로 검색, 이름 내림차순으로 */
	public List<Emp> selectByJobOrderByEnameDesc(String job) {
		JPAQuery<?> query = new JPAQuery<Void>(em);
		List<Emp> emps = query.select(emp).from(emp)
```

```
                                        .where(emp.job.eq(job))
                                        .orderBy(emp.ename.desc()).fetch();
        return emps;
}

@Override
@Transactional
/* job을 입력받아 EMP 삭제 */
public Long deleteByJob(String job) {
        Long affedtedRow = new JPADeleteClause(em, emp)
            .where(emp.job.eq(job))
            .execute();

        return affedtedRow;
}

/* 사번과 새이름을 입력받아 이름을 변경 */
@Override
@Transactional
public Long updateByEmpno(Long empno, String newEname) {
        Long affedtedRow = new JPAUpdateClause(em, emp)
        .where(emp.empno.eq(empno))
        .set(emp.ename, newEname)
        .execute();

        return affedtedRow;
}

/* job을 검색조건으로 ename, job 추출 */
@Override
public List<Tuple> selectEnameJobByEmpno(Long empno) {
        JPAQuery<?> query = new JPAQuery<Void>(em);

        //Multi Column Select
    List<Tuple> result = query.select(emp.ename, emp.job).from(emp)
                                .where(emp.empno.eq(empno))
                                .fetch();

        return result;
```

```
}

/* Emp, Dept를 조인하여 입력받은 부서의 사원명,부서명을 추출하는데 부서코드가
   없는 사원은 추출되지 않는다 */
@Override
public List<Tuple> selectEmpEnameDnameJoinDept(Long deptno) {
        JPAQuery<?> query = new JPAQuery<Void>(em);

        List<Tuple> emps = query.select(emp.ename, dept.dname).from(emp)
                        .innerJoin(emp.dept, dept)
                        .where(emp.dept.deptno.eq(deptno))
                        .fetch();

        return emps;
}

/* Emp 테이블에서 최대급여 사원 추출, 서브쿼리 */
@Override
public List<Emp> selectEmpMaxSal() {
        JPAQuery<?> query = new JPAQuery<Void>(em);
        QEmp e = new QEmp("e");

        List<Emp> emps = query.select(emp).from(emp)
                        .where(emp.sal.eq(
                                        JPAExpressions.select(e.sal.max()).from(e)))
                        .fetch();

        return emps;
}

/* 부서별 최대급여받는 사원 추출 , 서브쿼리 */
@Override
public List<Emp> selectEmpMaxSalOfDept() {
        JPAQuery<?> query = new JPAQuery<Void>(em);
        QEmp e = new QEmp("e");

        List<Emp> emps = query.select(emp).from(emp)
                                .where(emp.sal.eq(
                                        JPAExpressions
```

```
                                                .select(e.sal.max()).from(e)


            .where(emp.dept.deptno.eq(e.dept.deptno))
                                                ))

                                .fetch();

            return emps;
}

/* 자신이 속한 부서의 평균급여보다 급여가 많은 사원추출 ,서브쿼리 */
@Override
public List<Emp> selectEmpGreaterThanAvgSal() {
            JPAQuery<?> query = new JPAQuery<Void>(em);
            QEmp e = new QEmp("e");

            List<Emp> emps = query.select(emp).from(emp)
                                .where(emp.sal.gt(
                                                JPAExpressions
                                                .select(e.sal.avg()).from(e)


            .where(emp.dept.deptno.eq(e.dept.deptno))
                                                ))

                                .fetch();

            return emps;
}

/* 입력받은 사원과 급여가 같은 사원추출 , 서브쿼리 */
/* 입력받은 사원은 출력안함                          */
@Override
public List<Emp> selectEmpEqualsEmpno(Long empno) {
            JPAQuery<?> query = new JPAQuery<Void>(em);
            QEmp e = new QEmp("e");

            List<Emp> emps = query.select(emp).from(emp)
```

```
                .where(emp.sal.eq(
                                JPAExpressions
                                        .select(e.sal).from(e)

                                        .where(e.empno.eq(empno))
                                ))
                .where(emp.empno.ne(empno))
                .fetch();

        return emps;
}

/* Emp 테이블에서 급여상위 3명 추출 , 서브쿼리 */
@Override
public List<Emp> selectEmpMaxSalTop3() {

        JPAQuery<?> query = new JPAQuery<Void>(em);

        List<Emp> emps = query.select(emp).from(emp)
                .orderBy(emp.sal.desc())
                .limit(3)
                .fetch();

        return emps;
}

/* Dept 테이블에서 사원이 한명이라도 존재하는 부서명추출, 서브쿼리 */
@Override
public List<String> selectDeptExistsEmp() {
        JPAQuery<?> query = new JPAQuery<Void>(em);

        List<String> depts = query.select(dept.dname).from(dept)
                .where(JPAExpressions
                                .selectFrom(emp)


        .where(emp.dept.deptno.eq(dept.deptno)).exists()
                )
                .fetch();
```

```
                return depts;
        }
}
```

[EmpRepositoryQuery.java]

```
package jpa.repository;

import java.util.List;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.Repository;
import org.springframework.data.repository.query.Param;
import jpa.model.Emp;

// Spring Data JPA @Query지원 메소드
public interface EmpRepositoryQuery extends Repository<Emp, Long>
{
        //Native SQL, SQL구문은 JPQL형태가 아니라 DB에서 사용하는 SQL형식을 쓰면 된다.
        //nativeQuery 값의 default는 false
        //#entityName은 SpEL 표현이며 위 Repository<Emp, Long>의 Emp객체를 가리킨다.
        //위 선언에서 EmpRepositoryQuery<Emp> Repository<T, Long> 형태로 쓸 때 유용하다.
        @Query(value="select * from #{#entityName} e where e.ename = ?1",
                        nativeQuery=true)
        List<Emp> findByEname(String ename);

        @Query(value="select ename, job, sal from Emp e where e.sal > ?1 and e.sal < ?2 ")
        List<Emp> findBySalRange(Long sal1, Long sal2);

        //기본적으로 Spring Data JPA의 파라미터 바인딩은 순서에 의존한다.
        //쿼리의 바인딩 될 파라미터 이름으로 바인딩 하는 경우 @Param을 쓰면 된다.
        @Query(value="select ename, job, sal from Emp e where e.ename like %:ename% ")
        List<Emp> findByEnameMatch(@Param("ename") String ename);

        @Query(value="select ename, job, sal from Emp e where e.ename = :ename and e.job=:job and e.sal = :sal")
        List<Emp> findByNamedParam(@Param("ename") String ename,
                                @Param("job") String job,
```

```
                                @Param("sal") Long sal);

        @Query(value = "select e from Emp e where e.sal = :sal")
        List<Emp> findBySalCustom(@Param("sal") Long sal);

}
```

[EmpService.java]

```
package jpa.service;

import java.util.List;
import org.springframework.data.domain.Page;
import com.querydsl.core.Tuple;

import jpa.model.Dept;
import jpa.model.Emp;

public interface EmpService {
        List<Emp> findAll();
        void saveEmp(Emp emp);
        Emp findOne(Long empno);
        void delete(Long empno);

        //NamedQuery
        List<Emp> findBySalNamed(Long sal);

        //Query Method(메소드명으로 쿼리자동작성)
        List<Emp> findByEname(String ename);
        List<Emp> findBySalCustom(Long sal);
        List<Emp> findBySalRange(Long sal1, Long sal2);
        List<Emp> findByEnameMatch(String ename);
        List<Emp> findByNamedParam(String ename, String job, Long sal);
        List<Emp> findFirst3BySalBetweenOrderByEnameDesc(Long sal1, Long sal2);

        //Querydsl
        List<Emp> selectByJobOrderByEnameDesc(String job);
        Long deleteByJob(String job);
        Long updateByEmpno(Long empno, String newEname);
        List<Tuple> selectEnameJobByEmpno(Long empno);
```

```
        List<Tuple> selectEmpEnameDnameJoinDept(Long deptno);
        List<Emp> selectEmpMaxSal();
        List<Emp> selectEmpMaxSalOfDept();
        List<Emp> selectEmpGreaterThanAvgSal();
        List<Emp> selectEmpEqualsEmpno(Long empno);
        List<Emp> selectEmpMaxSalTop3();
        List<String> selectDeptExistsEmp();

        //Pagination
        Page<Emp> getEmpBySalGreaterThan(Long sal, Integer pageSize, Integer pageNumber);
}
```

[EmpServiceImpl.java]

```
package jpa.service;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;
import com.querydsl.core.Tuple;
import jpa.model.Dept;
import jpa.model.Emp;
import jpa.repository.EmpRepository;
import jpa.repository.EmpRepositoryQuery;

@Service("empService")
public class EmpServiceImpl implements EmpService {

        @Autowired
        private EmpRepository empRepository;

        @Autowired
        private EmpRepositoryQuery empRepositoryQuery;

    ////////////////////////// JpaRepository 기본 CRUD 메소드
        @Override
        public List<Emp> findAll() {
```

```
                //JpaRepository기본 메소드
                return empRepository.findAll();
        }

        @Override
        public void saveEmp(Emp emp) {
                //JpaRepository기본 메소드
                empRepository.save(emp);
        }

        @Override
        public Emp findOne(Long empno) {
                //JpaRepository기본 메소드
                return empRepository.findOne(empno);
        }

        @Override
        public void delete(Long empno) {
                //JpaRepository기본 메소드
                empRepository.delete(empno);
        }

        /////////////// EmpRepository에 만든 메소드(NamedQuery 호출)
        @Override
        public List<Emp> findBySalNamed(Long sal) {
                return empRepository.findBySalNamed(sal);
        }

        /////////////// EmpRepository에 만든 메소드(페이지처리용 Query Method 호출)
        /* 입력받은 sal값보다 크거나 같은사원 추출, 페이징 처리 */
        /* Query Method 호출                                    */
        @Override
        public Page<Emp> getEmpBySalGreaterThan(Long sal, Integer pageSize, Integer
pageNumber) {
        PageRequest request = new PageRequest(pageNumber - 1, pageSize, Sort.Direction.DESC,
"ename");
        return empRepository.findBySalGreaterThan(sal, request);   //1000:sal
    }
```

```
/* 급여로 조회하는데 이름내림차순으로 처음3건만 추출 */
/* Query Method 호출                                  */
@Override
public List<Emp> findFirst3BySalBetweenOrderByEnameDesc(Long sal1, Long sal2) {

        return empRepository.findFirst3BySalBetweenOrderByEnameDesc(sal1, sal2);
}

////////////////////////////////////// @Query 메소드
@Override
public List<Emp> findByEname(String ename) {
        return empRepositoryQuery.findByEname(ename);
}

@Override
public List<Emp> findBySalRange(Long sal1, Long sal2) {
        return empRepositoryQuery.findBySalRange(sal1, sal2);
}

@Override
public List<Emp> findByEnameMatch(String ename) {
        return empRepositoryQuery.findByEnameMatch(ename);
}

@Override
public List<Emp> findByNamedParam(String ename, String job, Long sal) {
        return empRepositoryQuery.findByNamedParam(ename, job, sal);
}

@Override
public List<Emp> findBySalCustom(Long sal) {
        return empRepositoryQuery.findBySalCustom(sal);
}

////////////////////////////////////// Querydsl 메소드
@Override
public List<Emp> selectByJobOrderByEnameDesc(String job) {
        return empRepository.selectByJobOrderByEnameDesc(job);
}
```

```
@Override
public Long deleteByJob(String job) {
	return empRepository.deleteByJob(job);
}

@Override
public Long updateByEmpno(Long empno, String newEname) {
	return empRepository.updateByEmpno(empno, newEname);
}

@Override
public List<Tuple> selectEnameJobByEmpno(Long empno) {
	return empRepository.selectEnameJobByEmpno(empno);
}

@Override
public List<Tuple> selectEmpEnameDnameJoinDept(Long deptno) {
	return empRepository.selectEmpEnameDnameJoinDept(deptno);
}

@Override
public List<Emp> selectEmpMaxSal() {
	return empRepository.selectEmpMaxSal();
}

@Override
public List<Emp> selectEmpMaxSalOfDept() {
	return empRepository.selectEmpMaxSalOfDept();
}

@Override
public List<Emp> selectEmpGreaterThanAvgSal() {
	return empRepository.selectEmpGreaterThanAvgSal();
}

@Override
public List<Emp> selectEmpEqualsEmpno(Long empno) {
	return empRepository.selectEmpEqualsEmpno(empno);
```

```
	}

	@Override
	public List<Emp> selectEmpMaxSalTop3() {
		return empRepository.selectEmpMaxSalTop3();
	}

	@Override
	public List<String> selectDeptExistsEmp() {
		return empRepository.selectDeptExistsEmp();
	}
}
```

[DeptService.java]

```
package jpa.service;

import jpa.model.Dept;

public interface DeptService {
	Dept findByDname(String dname);   //query method
	Dept saveDept(Dept dept);
}
```

[DeptServiceImpl.java]

```
package jpa.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import jpa.model.Dept;
import jpa.repository.DeptRepository;

@Service("deptService")
public class DeptServiceImpl implements DeptService {

	@Autowired
	private DeptRepository deptRepository;

	@Override
```

```
	public Dept findByDname(String dname) {
		Dept depts = deptRepository.findByDname(dname);
		return depts;
	}

	@Override
	public Dept saveDept(Dept dept) {
		return deptRepository.save(dept);
	}
}
```

[EmpController.java]

```
package jpa.controller;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.querydsl.core.Tuple;

import jpa.model.Dept;
import jpa.model.Emp;
import jpa.model.QDept;
import jpa.model.QEmp;
import jpa.service.DeptService;
import jpa.service.EmpService;

@RestController
@RequestMapping("/emp")
public class EmpController {
	@Autowired
	private EmpService empService;
```

```
	@Autowired
	private DeptService deptService;

	////////////////////////////// JpaRepository의 기본 CRUD
	/* localhost:8080/emp/add/홍길동/교수/9999/교육부 */
	@RequestMapping(value = "/add/{ename}/{job}/{sal}/{dname}")
	public Emp addEmp(@PathVariable String ename, @PathVariable String job,
@PathVariable Long sal,
			@PathVariable String dname) {

		Dept dept = deptService.findByDname(dname);
		if (dept == null) {
			dept = deptService.saveDept(new Dept(dname));
		}

		Emp emp = new Emp();
		emp.setEname(ename);
		emp.setJob(job);
		emp.setSal(sal);
		emp.setDept(dept);

		empService.saveEmp(emp);

		return emp;
	}

	////////////////////////////// JpaRepository의 기본 CRUD
	/* localhost:8080/emp/delete/9999 */
	@RequestMapping(value = "/delete/{empno}")
	public void deleteEmp(@PathVariable Long empno) {
		empService.delete(empno);
	}

	////////////////////////////// JpaRepository의 기본 CRUD
	/* localhost:8080/emp/ */
	@RequestMapping(value = "/")
	public List<Emp> findAll() {
		return empService.findAll();
	}
```

```
        ////////////////////////// JpaRepository의 기본 CRUD
        /* localhost:8080/emp/1 */
        @RequestMapping(value = "/{empno}")
        public Emp findOne(@PathVariable Long empno) {
                return empService.findOne(empno);
        }

        ////////////////////////// Named Query
        @RequestMapping(value = "/search/sal/{sal}")
        public List<Emp> findBySalNamed(@PathVariable Long sal) {
                return empService.findBySalNamed(sal);
        }

        ////////////////////////// @Query
        /* localhost:8080/emp/search/ename/홍길동 */
        @RequestMapping(value = "/search/ename/{ename}")
        public List<Emp> findByEname(@PathVariable String ename) {
                return empService.findByEname(ename);
        }

        ////////////////////////// @Query
        /* localhost:8080/emp/search/custom/sal/9999 */
        @RequestMapping(value = "/search/custom/sal/{sal}")
        public List<Emp> findBySalCustom(@PathVariable Long sal) {
                return empService.findBySalCustom(sal);
        }

        ////////////////////////// @Query
        @RequestMapping(value = "/search/custom/top3/{sal1}/{sal2}")
        public List<Emp> findFirst3BySalBetweenOrderByEnameDesc(@PathVariable Long sal1,
@PathVariable Long sal2) {
                return empService.findFirst3BySalBetweenOrderByEnameDesc(sal1, sal2);
        }

        ////////////////////////// @Query
        @RequestMapping(value = "/search/match/ename/{ename}")
        public List<Emp> findByEnameMatch(@PathVariable String ename) {
                return empService.findByEnameMatch(ename);
```

```
	}

	////////////////////////// @Query
	/* localhost:8080/emp/search/param/홍길동/교수/9999 */
	@RequestMapping(value = "/search/param/{ename}/{job}/{sal}")
	public List<Emp> findByNamedParam(@PathVariable String ename, @PathVariable String job, @PathVariable Long sal) {
		return empService.findByNamedParam(ename, job, sal);
	}

	////////////////////////// @Query
	/* localhost:8080/emp/search/sal/5555/9999 */
	@RequestMapping(value = "/search/sal/{sal1}/{sal2}")
	public List<Emp> findBySalRange(@PathVariable Long sal1, @PathVariable Long sal2) {
		return empService.findBySalRange(sal1, sal2);
	}

	////////////////////////// @Query
	// localhost:8080/emp/search/sal/1000/3/2
	// SAL이 1000 보다 크거나 같은데, 한페이지3개씩,두번째페이지
	@RequestMapping(value = "/search/sal/{sal}/{pageSize}/{pageNumber}")
	public List<Emp> getEmpBySalGreaterThan(@PathVariable Long sal, @PathVariable Integer pageSize,
			@PathVariable Integer pageNumber) {
		Page<Emp> result = empService.getEmpBySalGreaterThan(sal, pageSize, pageNumber);
		List<Emp> emps = result.getContent();
		return emps;
	}

	////////////////////////// Querydsl
	/* localhost:8080/emp/search/job/교수 */
	@RequestMapping(value = "/search/job/{job}")
	public List<Emp> getEmpBySal(@PathVariable String job) {
		return empService.selectByJobOrderByEnameDesc(job);
	}

	////////////////////////// Querydsl
	/* localhost:8080/emp/delete/job/교수 */
```

```
	@RequestMapping(value = "/delete/job/{job}")
	public String deleteEmpByJob(@PathVariable String job) {
		Long affectedRow = empService.deleteByJob(job);
		return "[job:" + job + "] " + affectedRow + " row deleted!";
	}

	////////////////////////// Querydsl
	/* localhost:8080/emp/update/empno/1/1길동 */
	@RequestMapping(value = "/update/empno/{empno}/{newEname}")
	public String updateEmpByEmpno(@PathVariable Long empno, @PathVariable String newEname) {
		Long affectedRow = empService.updateByEmpno(empno, newEname);
		return "[empno:" + empno + "] " + affectedRow + " row updated!";
	}

	////////////////////////// Querydsl(다중칼럼 선택)
	/* localhost:8080/emp/select/empno/1 */
	@RequestMapping(value = "/select/empno/{empno}")
	public Map<String, String> selectEmpEnameSalByJob(@PathVariable Long empno) {
		Map<String, String> m = new HashMap<String, String>();
		QEmp emp = QEmp.emp;
		List<Tuple> result = empService.selectEnameJobByEmpno(empno);
		for (Tuple row : result) {
			m.put(row.get(emp.ename), row.get(emp.job));
		}
		return m;
	}

	////////////////////////// Querydsl(join)
	/* localhost:8080/emp/select/enamedname/1 */
	/* 사원, 부서를 조인하여 사원이름, 부서명 추출 */
	@RequestMapping(value = "/select/enamedname/{deptno}")
	public Map<String, String> getEmpEnameDnameJoinDept(@PathVariable Long deptno) {
		Map<String, String> m = new HashMap<String, String>();
		QEmp emp = QEmp.emp;
		QDept dept = QDept.dept;
		List<Tuple> result = empService.selectEmpEnameDnameJoinDept(deptno);
		for (Tuple row : result) {
			m.put(row.get(emp.ename), row.get(dept.dname));
```

```
            }
            return m;
    }

    ////////////////////////// Querydsl(sub query)
    /* localhost:8080/emp/select/maxsal */
    /* 최대급여를 가지는 사원 추출 */
    @RequestMapping(value = "/select/maxsal")
    public List<Emp> selectEmpMaxSal() {
            return empService.selectEmpMaxSal();
    }

    ////////////////////////// Querydsl(sub query)
    /* localhost:8080/emp/select/emp/maxsalofdept */
    /* 부서별 최대 급여사원 출력 */
    @RequestMapping(value = "/select/maxsalofdept")
    public List<Emp> selectEmpMaxSalOfDept() {
            return empService.selectEmpMaxSalOfDept();
    }

    ////////////////////////// Querydsl(sub query)
    /* localhost:8080/emp/select/emp/gt/avgsalofdept */
    /* 자신이 속한 부서의 평균급여보다 급여가 많은 사원 */
    @RequestMapping(value = "/select/gt/avgsalofdept")
    public List<Emp> selectEmpGreaterThanAvgSal() {
            return empService.selectEmpGreaterThanAvgSal();
    }

    ////////////////////////// Querydsl(sub query)
    /* localhost:8080/emp/select/same/1 */
    /* 입력받은 사원과 급여가 같은 사원 추출, 입력받은 사원은 제외 */
    @RequestMapping(value = "/select/same/{empno}")
    public List<Emp> selectEmpEqualsEmpno(@PathVariable Long empno) {
            return empService.selectEmpEqualsEmpno(empno);
    }

    ////////////////////////// Querydsl(sub query)
    /* localhost:8080/emp/select/top3 */
    /* Emp 테이블에서 급여 상위 3명 추출 */
```

```
	@RequestMapping(value = "/select/top3")
	public List<Emp> selectEmpMaxSalTop3() {
		return empService.selectEmpMaxSalTop3();
	}

	////////////////////////// Querydsl(sub query)
	/* localhost:8080/emp/select/exists */
	/* 사원이 한명이라도 존재하는 부서명 추출 */
	@RequestMapping(value = "/select/exists")
	public List<String> selectDeptExistsEmp() {
		return empService.selectDeptExistsEmp();
	}
}
```