Spring Data JPA, QueryDSL실습 – JPAQueryFactory 이용

■ 이전 예제의 QueryDSL 작성 부분만 JPAQueryFactory를 사용하여 구현해보자. 다른점은 JPAQueryFactory 인스턴스 생성을 위해 스프링 부트 메인에 @Bean으로 정의한 부분과 EmpRepositoryImpl의 쿼리 작성 부분이 조금 다르고 groupBy 처리 메소드가 추가되었다.

STS -> Spring Starter Project
  project name : jpaqueryfactory-exam
  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

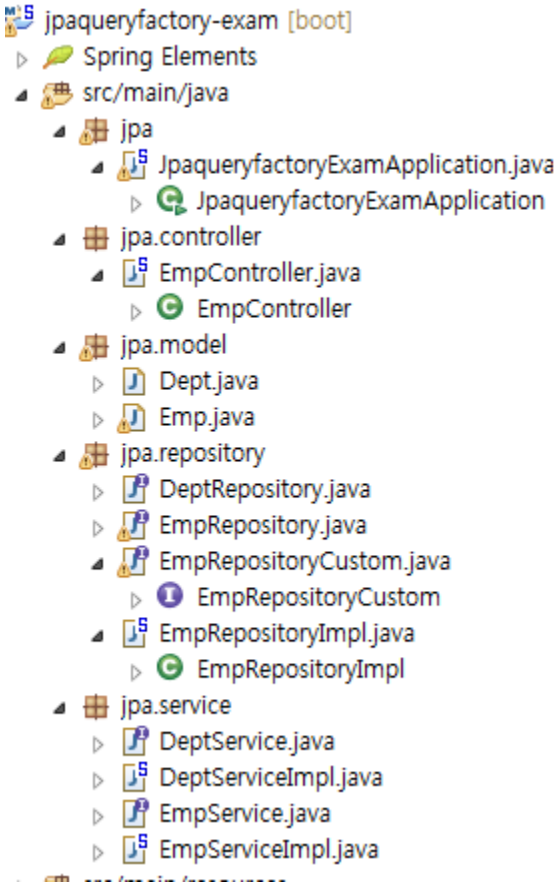
[pom.xml]

```
<?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.edu</groupId>
        <artifactId>springdatajpa_querydsl</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>
```

```
<name>jpaqueryfactory-exam</name>
<description>jpaqueryfactory example</description>

<parent>
	<groupId>org.springframework.boot</groupId>
	<artifactId>spring-boot-starter-parent</artifactId>
	<version>1.3.3.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.projectlombok</groupId>
		<artifactId>lombok</artifactId>
		<version>1.16.6</version>
	</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>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>

        </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>
```



[JpaqueryfactoryExamApplication.java]

```
package jpa;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
```

```
import com.querydsl.jpa.impl.JPAQueryFactory;

@SpringBootApplication
public class JpaqueryfactoryExamApplication {

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

	@PersistenceContext
	EntityManager em;

	@Bean
	public JPAQueryFactory queryFactory() {
		//return new JPAQueryFactory(JPQLTemplates.DEFAULT, em);
		return new JPAQueryFactory(em);
	}
}
```

[application.properties]

```
spring.datasource.platform=mysql
spring.datasource.sql-script-encoding=UTF-8
spring.datasource.url=jdbc:mysql://localhost/queryfactoryexam?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
public class Dept   {
        @Id
        @GeneratedValue
        private Long deptno;

        @Column(unique=true)
        private String dname;

        public Dept(String dname) {
                this.dname = 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 javax.persistence.Tuple;
import org.springframework.data.jpa.repository.JpaRepository;
```

```
import jpa.model.Emp;
/*
 * EmpRepositoryCustom을 상속
 */
public interface EmpRepository extends JpaRepository<Emp, Long>, EmpRepositoryCustom{

}
```

[EmpRepositoryCustom.java] – 사용자 정의 레포지토리

```
package jpa.repository;



import java.util.List;
import com.querydsl.core.Tuple;
import jpa.model.Emp;
/*
 * 사용자 정의 인터페이스, 기본으로 제공되는 JpaRepository이외의
 * 사용자 쿼리 작성할 때 만드는 인퍼테이스이며 QueryDSL을 휘한 메소드 정의
*/
public interface EmpRepositoryCustom {
	//QueryDSL 처리용
	List<Emp> selectByJobOrderByEnameDesc(String job);
	Long deleteByJob(String job);
	Long updateByEmpno(Long empno, String newEname);
	List<Tuple> selectEnameJobByEmpno(Long empno);
	List<Tuple> selectSalAvgGroupbyJob(Long sumSal);
	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 org.springframework.beans.factory.annotation.Autowired;
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.JPAQueryFactory;
import jpa.model.Emp;
import jpa.model.QEmp;

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

	@Autowired
	JPAQueryFactory queryFactory;

	////////////////////////////////// QueryDSL용 메소드

	@Override
	/* Emp 테이블에서 job을 조건으로 검색, 이름 내림차순으로 */
	public List<Emp> selectByJobOrderByEnameDesc(String job) {

		List<Emp> emps = queryFactory.selectFrom(emp)
					.where(emp.job.eq(job))
					.orderBy(emp.ename.desc()).fetch();
		return emps;
	}

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

```
//                      .execute();
                    Long affectedRow = queryFactory.delete(emp)
                        .where(emp.job.eq(job))
                        .execute();

                    return affectedRow;
            }

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

                    return affectedRow;
            }

            /* job을 검색조건으로 ename, job 추출 */
            @Override
            public List<Tuple> selectEnameJobByEmpno(Long empno) {
                    //Multi Column Select
                List<Tuple> result = queryFactory.select(emp.ename, emp.job)
                                                 .from(emp)
                                                 .where(emp.empno.eq(empno))
                                                 .fetch();

                    return result;
            }

            /* Emp 테이블에서 job별로 GroupBy후 그룹별 급여평균추출
               단 그룹의 급여 합계가 주어진 합보다 큰경우 */
            @Override
```

```
public List<Tuple> selectSalAvgGroupbyJob(Long sumSal) {
        List<Tuple> result = queryFactory
                        .select(emp.job, emp.sal.avg())
            .from(emp)
            .groupBy(emp.job)
            .having(emp.sal.sum().gt(sumSal))
            .fetch();

        return result;
}

/* Emp 테이블에서 입력받은 부서원 이름 및 부서명을 추출하는데 Dept 테이블과 조인
   부서코드를 안가지는 사원은 추출되지 않는다 */
@Override
public List<Tuple> selectEmpEnameDnameJoinDept(Long deptno) {
        List<Tuple> emps = queryFactory
                        .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() {
        QEmp e = new QEmp("e");

        List<Emp> emps = queryFactory.selectFrom(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() {
	QEmp e = new QEmp("e");

	List<Emp> emps = queryFactory.selectFrom(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() {
	QEmp e = new QEmp("e");

	List<Emp> emps = queryFactory.selectFrom(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) {
```

```
        QEmp e = new QEmp("e");

        List<Emp> emps = queryFactory.selectFrom(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() {
        List<Emp> emps = queryFactory.selectFrom(emp)
                .orderBy(emp.sal.desc())
                .limit(3)
                .fetch();

        return emps;
    }

    /* Dept 테이블에서 사원이 한명이라도 존재하는 부서명추출, 서브쿼리 */
    @Override
    public List<String> selectDeptExistsEmp() {
        List<String> depts = queryFactory.select(dept.dname).from(dept)
                .where(JPAExpressions
                        .selectFrom(emp)

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

        return depts;
    }
```

```
}
```

[EmpService.java]

```
package jpa.service;

import java.util.List;

import com.querydsl.core.Tuple;

import jpa.model.Emp;

public interface EmpService {
	//JpaRepositiry 기본 CRUD
	void saveEmp(Emp emp);

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

[EmpServiceImpl.java]

```
package jpa.service;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.querydsl.core.Tuple;
import jpa.model.Emp;
```

```
import jpa.repository.EmpRepository;

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

	@Autowired
	private EmpRepository empRepository;

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

	////////////////////////////////////// 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> selectSalAvgGroupbyJob(Long sumSal) {
		return empRepository.selectSalAvgGroupbyJob(sumSal);
	}
```

```
	@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.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;

	////////////////////////// JpaRepository기본 CRUD 메소드
	/* 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;
        }

        /////////////////////////// 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(GroupBy)
	/* localhost:8080/emp/select/job/groupby/60000 */
	/* Emp 테이블에서 job별로 GroupBy후 그룹별 급여평균추출
	   단 그룹의 급여 합계가 주어진 합보다 큰경우 */
	@RequestMapping(value = "/select/job/groupby/{sumSal}")
	public Map<String, Double> selectSalAvgGroupbyJob(@PathVariable Long sumSal) {
		Map<String, Double> m = new HashMap<String, Double>();
		QEmp emp = QEmp.emp;
		List<Tuple> result = empService.selectSalAvgGroupbyJob(sumSal);
		for (Tuple row : result) {
			m.put(row.get(emp.job), row.get(emp.sal.avg()));
		}
		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();
        }
}
```