6/3 백엔드

### Eager Loading 정책의 장점

어차피 시장에 왔으니, 필요할 것 같은 상품들을 다 사오는 것이 편하다.

어차피 DB에 연결해서 명령을 실행하고 데이터를 받아오고 있으니, 필요할 것 같은 데이터들은 한 번에 다 가져오는 것이 좋다.

### Eager Loading 정책의 단점

사온 상품이 필요 없다는 것을 알게 되면 돈 낭비다.

가져온 데이터를 결국 사용하지 않게 된다면, 자원 낭비다.

### Lazy Loading 정책의 장점

비록 지금 시장에 왔지만, 현재 필요하다는 확신이 없는 상품을 사지 않는다.

학과 레코드를 조회해서 Department 엔터티 객체에 채울 때,

그 학과 소속 학생 목록 레코드들을 조회하지 않는다.

돈 낭비, 자원 낭비를 줄인다.

### Lazy Loading 정책의 단점

아까 시장에서, 확신이 없어서 사지 않았던 상품이, 지금 필요하면

다시 시장에 가서 사와야 한다. 시간이 걸린다.

그 학과 소속 학생 목록이 필요하면,

또 DB에 연결해서 그 학과 소속 학생 레코드들을 조회해야 한다.

시간이 걸린다.

departments 는 액션 메소드가 전달한 model attribute 이름이다.

departments 학과 목록의 학과 각각을 dp 변수에 대입하면서

tr 태그와 그 아래 td 태그들이 반복 출력된다.

id, name, shortName, phone 부분은 학과(Department) 객체의 속성이다.

RestController를 구현한다면,

@OneToMany 속성에 @JsonIgnore를 붙여야 한다.

엔터티 객체의 toString 메소드를 호출할 거라면,

@OneToMany 속성에 @ToString.Exclude, @EqualsAndHashCode.Exclude를 붙여야 한다.

그런데, 디버깅을 위한 로그 기록에서,

엔터티 객체의 toString 메소드가, 나도 모르게 호출될 수 있다.

@OneToMany 속성에 @JsonIgore과 @ToString.Exclude를 언제나 붙이자.

## @Table, @Column

JPA 엔터티 클래스 이름과 DB 테이블 이름이 일치해야 한다.

다행히 Windows mysql은 대소문자에 관대해서, 대소문자가 일치하지 않아도 되지만,

linux mysql은 대소문자까지 일치해야 한다.

student2 데이터베이스의 학생 테이블은 student 이고

학생 엔터티 클래스 이름은 Student 이다.

대소문자가 다르기 때문에 linux mysql에서 실행하면 에러가 발생한다.

만약 이름이 일치하지 않은 경우에,

@Table(name="테이블이름") 어노테이션을 사용해서 테이블 이름을 명시해 줄 수 있다.

JPA 엔터티 클래스의 속성 이름이 DB 테이블의 필드 이름과 일치해야 한다.

만약 이름이 일치하지 않은 경우에,

@Column(name="필드이름") 어노테이션을 사용해서 필드 이름을 명시해 줄 수 있다.

1) department id 값이 3, 4 인 학생들만 조회하기

메소드 호출: studentRepository.findByDepartmentIdOrDepartmentId(3, 4);

2) 김씨 이거나 박씨인 학생들 조회하기

메소드 호출: studentRepository.findByNameStartsWithOrNameStartsWidth("김", "박");

3) 김씨 학생을 이름 오름차순으로 10명만 조회하기

메소드 호출: studentRepository.findTop10ByNameStartsWidthOrderByName("김");

4) 김씨인 소프학생들만 이름 오름차순으로 조회하기

메소드 호출: studentRepository.findByNameStartsWithAndDepartmentId("김", 1);

학생을 조회하는 메소드들이니까 StudentRepository에 구현해야함.

@RequestParam(value="departmentId", required=false, defaultValue="0") int departmentId

departmentId 이름의 request parameter를 전달 받기 위한 변수를 선언한다.

value="departmentId"

request parameter 이름과 일치해야 한다.

즉 <input type="text" name="departmentId"> 부분과 일치해야 한다.

request=false

이 이름의 request parameter가 없어도 에러가 발생하지 않는다는 옵션.

이 옵션이 없고 이 이름의 request parameter도 없으면 에러가 발생한다.

조회 조건 유지 구현

model.addAttribute("departmentId", departmentId);

select 태그에서 선택된 값 즉 departmentId 값이 뷰에 전달되어야 한다.

th:selected="${d.id == departmentId}

@JoinColumn(name = "departmentId")

departmentId는 student 테이블의 외래키인 departmentId 필드명과 일치.

@OneToMany(mappedBy="department")

department는 이 관계의 저쪽 Student 엔터티 클래스의 department 속성명과 일치.

### Professor

|  |
| --- |
| @ManyToOne  @JoinColumn(name = "departmentId")  Department department; |

### Department

|  |
| --- |
| @OneToMany(mappedBy="department")  List<Professor> professors; |

List<Student> findBySugangsLectureTitle(String title);

Student 엔터티에는 sugangs 속성이 있고, 이 속성의 타입은 List<Sugang> 이다.

Sugang 엔터티에는 lecture 속성이 있고, 이 속성의 타입은 Lecture 이다.

Lecture 엔터티네는 title 속성이 있고, 이 속성의 타입은 String 이다.