JPA 기초

2020-11-12

이승진

**학습목표**

JPA 기초 개념 및 구현 방법

RestController 구현

**목차**

[1. 배경 지식 2](#_Toc56239273)

[1) REST API 2](#_Toc56239274)

[2) JSON (Javascript Object Notation) 2](#_Toc56239275)

[3) RestController 2](#_Toc56239276)

[4) @ResponseBody 2](#_Toc56239277)

[5) @RequestBody 3](#_Toc56239278)

[6) Request Method 3](#_Toc56239279)

[7) REST API URL 패턴 4](#_Toc56239280)

[8) @PathVriable 4](#_Toc56239281)

[9) 액션 메소드 URL 4](#_Toc56239282)

[10) JPA Entity 클래스 5](#_Toc56239283)

[11) JPA Repository 기본 메소드 6](#_Toc56239284)

[2. jpa1 프로젝트 7](#_Toc56239285)

[1) student2 데이터베이스 import 7](#_Toc56239286)

[2) 프로젝트 생성 8](#_Toc56239287)

[3) pom.xml 10](#_Toc56239288)

[4) application.properties 10](#_Toc56239289)

[5) Department 엔터티 클래스 구현 11](#_Toc56239290)

[6) Department 리포지토리 구현 12](#_Toc56239291)

[7) Department 컨트롤러 구현 13](#_Toc56239292)

[3. 부록 - frontend 앱 16](#_Toc56239293)

[1) departments.css 16](#_Toc56239294)

[2) departments.html 17](#_Toc56239295)

# 배경 지식

## REST API

안드로이드 앱을 위한 서버를 개발하거나,

웹브라우저에서 실행되는 Javascript로 개발한 앱을 위한 서버를 개발할 때,

REST API 서비스를 제공하는 서버를 구현한다.

REST API 서비스라는 것은,

클라이언트의 URL 요청에 대해서 JSON 형태의 데이터를 출력하는 서버의 메소드를 말한다.

## JSON (Javascript Object Notation)

서버와 클라이언트 사이에 데이터를 주고 받을 때 흔히 사용되는 포멧이다.

JSON 포맷은 Javascript 언어의 객체나 배열의 문법과 동일하다.

JSON의 예는 다음과 같다.

### 객체

{ id:1, name:"홍길동", age:18 }

### int 배열

[1, 2, 3, 4]

### 객체배열

[{ id:1, name:"홍길동", age:18 }, { id:2, name:"전우치", age:19}]

## RestController

Spinrg MVC 프레임웍에서 REST API 서비스를 구현할 때,

컨트롤러에 @RestController 어노테이션을 붙인다.

RestController의 액션 메소드가 리턴하는 Java 객체는 자동으로 JSON 포멧으로 변환되어

클라이언트에 전송된다.

RestController의 액션 메소드는, 데이터를 클라이언트에 전송하기 때문에, 뷰(view)가 필요 없다.

그래서 뷰의 이름을 리턴할 필요없고,

클라이언트에게 전송하기 위한 데이터를 리턴한다.

## @ResponseBody

컨트롤러의 액션 메소드 앞에 @ResponseBody 어노테이션을 붙이면,

이 액션 메소드가 리턴하는 Java 객체는 자동으로 JSON 포멧으로 변환되어

클라이언트에 전송된다.

예:

|  |
| --- |
| @Controller  public class StudentController {  @RequestMapping("students")  public @ResponseBody List<Student> students() {  List<Student> list = ...;  return list;  } |

그런데 컨트롤러 앞에 @RestController 어노테이션을 붙였다면,

그 컨트롤러의 액션 메소드 앞에 @ResponseBody 어노테이션을 붙이지 않고 생략해도 된다.

예:

|  |
| --- |
| @RestController  public class StudentController {  @RequestMapping("students")  public List<Student> students() {  List<Student> list = ...;  return list;  }  } |

@RestController 어노테이션을 붙인 컨트롤러 클래스의 모든 메소드는

자동으로 @ResponseBody 어노테이션이 적용된다.

즉 @RestController 클래스의 액션 메소드는 뷰의 이름을 리턴할 수 없고,

데이터 객체를 리턴해야 한다.

## @RequestBody

서버가 클라이언트에 전송하는 데이터 뿐만 아니라,

클라이언트가 서버에 전달하는 request parameter 데이터도 JSON 포멧인 것이 일관적이어서 좋다.

이렇게 JSON 포멧으로 전송된 request parameter 데이터를 받을 액션 메소드의 파라미터 변수에는

@RequestBody 어노테이션을 붙여주어야 한다.

예:

|  |
| --- |
| public String update(@RequestBody Student student) {  ...  } |

주의 사항: request parameter 데이터가 JSON 포멧이 아니라면 @RequestBody를 붙이지 않아야 한다.

## Request Method

클라이언트가 서버에 요청(request)을 전달하는 방식(method)을 request method 라고 한다.

지금까지 우리가 사용한 request method는 GET, POST 둘 뿐이다.

그런데 GET, POST 뿐만 아니라 DELETE, PUT 도 있다.

REST API 서비스를 구현할 때, request method 들을 전부 잘 활용하는 것이 바람직하다. (관례이다)

서버에 데이터를 요청할 때는 GET 방식

서버에 저장할 새 데이터를 전송할 때는 POST 방식

서버의 기존 데이터를 수정하기 위해 전송할 때는 PUT 방식

서버의 데이터 삭제를 요청할 때는 DELETE 방식을 사용한다.

## REST API URL 패턴

### query string 사용하지 않기

REST API 서비스의 URL에 query string을 사용하지 않는 것이 관례이다.

예를 들어 아래 URL은 바람직하지 않다.

http://localhost:8080/studentServer/api/student?id=3

아래와 같은 형태이어야 한다.

http://localhost:8080/studentServer/api/student/3

Request Method = GET

### 동사 사용하지 않기

REST API 서비스의 URL에 동사를 사용하지 않는 것이 관례이다.

예를 들어 아래 URL은 바람직하지 않다.

http://localhost:8080/studentServer/studentDelete?id=3

아래와 같은 형태이어야 한다.

http://localhost:8080/studentServer/api/student/3

Request Method = DELETE

## @PathVariable

요청된 URL이 아래와 같다면

http://localhost:8080/jpa10/api/student/3

URL에 들어있는 id 값 3을 받기 위한 액션 메소드는 다음과 같이 구현한다.

|  |
| --- |
| @RequestMapping("api/student/{id}")  public Student student(@PathVariable("id") int id) |

## 액션 메소드 URL

컨트롤러 클래스에도 @RequestMapping("URL1") 어노테이션 붙어있고

액션 메소드에도 @RequestMapping("URL2") 어노테이션이 붙어있다면,

그 액션 메소드를 호출하기 위한 URL은 "URL1**/**URL2" 이다.

예:

|  |
| --- |
| @Controller  @RequestMapping("student")  public class StudentController {  @RequestMapping("list")  public String list(...) {  ...  }  } |

list 액션 메소드를 호출하기 위한 URL은 "student/list" 이다.

## JPA (Java Persistence API)

대표적인 Java ORM 기술은 mybatis와 JPA 이다.

아직까지는 mybatis가 좀 더 많이 사용되지만,

미래의 대세는 JPA 이다.

## JPA Entity 클래스

mybatis의 DTO 클래스에 해당하는 것이 JPA Entity 클래스이다.

JPA Entity 클래스는, 데이터베이스 테이블의 레코드에 해당하는 Java 클래스이다.

데에터베이스 조회 결과가 자동으로 엔터티 클래스 객체에 채워져 리턴된다.

### @Entity

JPA Entity 클래스 앞에 @Entity 어노테이션을 붙여야 한다.

### @Id

기본키(primary key)에 해당하는 멤버 변수에 @Id 어노테이션을 붙여야 한다.

### @GeneratedValue(strategy = GenerationType.IDENTITY)

기본키가 Auto Increment 필드이거나 Identity 필드인 경우에

이 어노테이션을 붙여야 한다.

## JPA Repository 기본 메소드

mybatis의 mapper 인터페이스에 해당하는 것이 JPA Repository 인터페이스이다.

|  |
| --- |
| package net.skhu.repository;  import org.springframework.data.jpa.repository.JpaRepository;  import net.skhu.entity.Department;  public interface DepartmentRepository extends JpaRepository<Department, Integer> {  } |

위와 같은 형태로 DepartmentRepository 인터페이스를 선언하면,

이 인터페이스를 구현한 클래스를 spring JPA 가 자동으로 구현해 준다.

이렇게 자동으로 구현된 클래스에는 아래와 같은 기본 메소드가 포함되어 있다.

Department 엔티티 클래스 이름

Integer primary key의 타입임

### departmentRepository.findAll() 메소드

department 테이블에서 레코드 전체 목록을 조회한다.

List<Department> 객체가 리턴된다.

### departmentRepository.findById(id)

department 테이블에서 기본키 필드인 id 값으로 레코드를 조회한다.

Optional<Department> 타입의 객체가 리턴된다. 이 객체의 get 메소드를 호출하면 Department 객체가 리턴된다.

예: Department department = departmentRepository.findById(id).get();

### departmentRepository.save(department)

Department 객체를 department 테이블에 저장한다.

Department 객체의 id(기본키) 속성값이 0 이면 INSERT 되고, 0 이 아니면 UPDATE 된다.

### departmentRepository.saveAll(departmentList)

Department 객체 목록을 department 테이블에 저장한다.

### departmentRepository.delete(department)

Department 객체의 id(기본키) 값과 일치하는 레코드가 삭제된다.

### departmentRepository.deleteById(id)

id(기본키) 값과 일치하는 레코드가 삭제된다.

### departmentRepository.deleteAll(departmentList)

departmentList 목록에 들어있는 Department 객체 목록을 테이블에서 삭제한다.

### departmentRepository.deleteAll()

department 테이블의 모든 레코드들을 삭제한다.

### departmentRepository.count()

department 테이블의 전체 레코드 수를 리턴한다.

### departmentRepository.existsById(id)

department 테이블에서 id에 해당하는 레코드가 있는지 true/false를 리턴한다.

### departmentRepository.flush() - 이 메소드 없어진 듯

지금까지 department 테이블에 대한 데이터 변경 작업들이 디스크에 모두 기록되도록 한다.

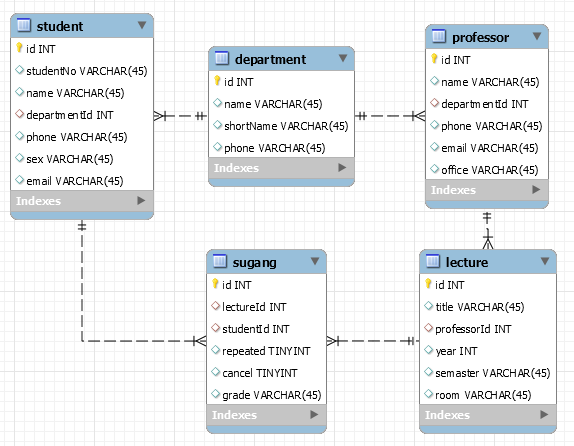
위 메소드들은 Spring JPA가 자동으로 구현해준 메소드이다.

# jpa1 프로젝트

## student2 데이터베이스 import

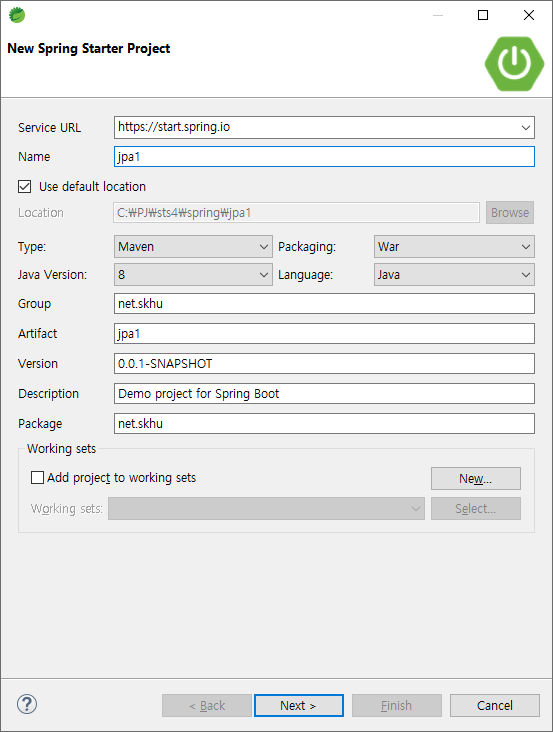
MySQL 설치 강의노트를 참고하여

student2 데이터베이스를 MySQL에 import 하자.



## 프로젝트 생성

메뉴: File - New - Spring Starter Project



|  |  |
| --- | --- |
| Name | jpa1 |
| Packaing | war |
| Group | net.skhu |
| Artifact | jpa1 |
| Package | net.skhu |

Graphical user interface, text, application

Description automatically generated

Developer Tools 항목 아래

Spring Boot DevTools

Lombok

SQL 항목 아래

Spring Data JPA

MySQL Driver

Web 항목 아래

Spring Web

## pom.xml

|  |
| --- |
| <dependency>  <groupId>org.apache.tomcat.embed</groupId>  <artifactId>tomcat-embed-jasper</artifactId>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  </dependency> |

## application.properties

**src/main/resources/application.properties**

|  |
| --- |
| spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  spring.datasource.url=jdbc:mysql://localhost:3306/student2?useUnicode=yes&characterEncoding=UTF-8&allowMultiQueries=true&serverTimezone=Asia/Seoul  spring.datasource.username=user1  spring.datasource.password=skhuA+4.5  spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl  logging.level.org.hibernate.SQL=DEBUG  logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE  server.port=8088 |

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/student2?useUnicode=yes&characterEncoding=UTF-8&allowMultiQueries=true&serverTimezone=Asia/Seoul

spring.datasource.username=user1

spring.datasource.password=test123

MySQL 서버의 student2 DB 연결 설정이다.

spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl

엔터티 클래스의 studentNo 속성에 자동으로 연결될 데이터베이스 필드 명이

studentNo 형태라면 위 설정이 필요하다. (camel case)

student\_no 형태이면 위 설정이 필요없다. (snake case)

camel case의 예: departmentManagerOfficeNumber

snake case의 예: department\_manager\_office\_number

logging.level.org.hibernate.SQL=DEBUG

logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE

JPA 엔진에 의해서 자동 생성되고 실행되는 SQL 명령을 실시간으로 이클립스 콘솔창에 보여준다.

## Department 엔터티 클래스 구현

### src/main/java/net/skhu/entity/Department.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | package net.skhu.entity;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import lombok.Data;  @Data  @Entity  public class Department {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  int id;  String name;  String shortName;  String phone;  } |

JPA Entity 클래스는, 데이터베이스 테이블의 레코드에 해당하는 Java 클래스이다.

데에터베이스 조회 결과가 자동으로 엔터티 클래스 객체에 채워져 리턴된다.

### @Data

Lombok 어노테이션이다.

Lombok에 의해서 getter, setter, equals, hashCode 메소드가 자동으로 구현된다.

### @Entity

JPA Entity 클래스 앞에 @Entity 어노테이션을 붙여야 한다.

### @Id

기본키(primary key)에 해당하는 멤버 변수에 @Id 어노테이션을 붙여야 한다.

### @GeneratedValue(strategy = GenerationType.IDENTITY)

기본키가 Auto Increment 필드이거나 Identity 필드인 경우에

이 어노테이션을 붙여야 한다.

## Department 리포지토리 구현

### src/main/java/net/skhu/repository/DepartmentRepository.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | package net.skhu.repository;  import org.springframework.data.jpa.repository.JpaRepository;  import net.skhu.entity.Department;  public interface DepartmentRepository extends JpaRepository<Department, Integer> {  } |

Integer 이 부분은 primary key의 타입이다.

## Department 컨트롤러 구현

### src/main/java/net/skhu/controller/DepartmentController.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | package net.skhu.controller;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.DeleteMapping;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.PostMapping;  import org.springframework.web.bind.annotation.PutMapping;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RestController;  import net.skhu.entity.Department;  import net.skhu.repository.DepartmentRepository;  @RestController  public class DepartmentController {  @Autowired DepartmentRepository departmentRepository;  @GetMapping("departments")  public List<Department> departments() {  return departmentRepository.findAll();  }  @GetMapping("department/{id}")  public Department department(@PathVariable("id") int id) {  return departmentRepository.findById(id).get();  }  @PostMapping("department")  public boolean insert(@RequestBody Department department) {  departmentRepository.save(department);  return true;  }  @PutMapping("department")  public boolean update(@RequestBody Department department) {  departmentRepository.save(department);  return true;  }  @DeleteMapping("department/{id}")  public boolean delete(@PathVariable("id") int id) {  departmentRepository.deleteById(id);  return true;  }  } |

DepartmentController 컨트롤러는 @RestController 이기 때문에 뷰(view)가 필요없다.

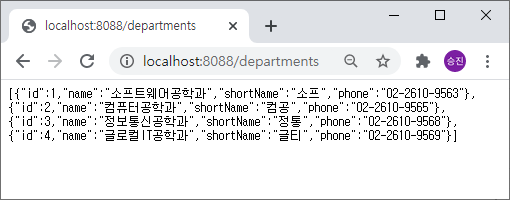
액션 메소드가 리턴하는 Java 객체가 자동으로 JSON 포멧으로 변환되어 출력된다.

|  |
| --- |
| @GetMapping("departments")  public List<Department> departments() {  return departmentRepository.findAll();  } |

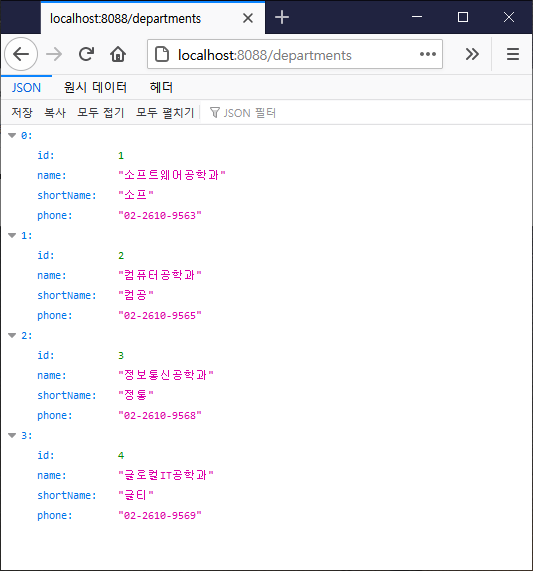
http://localhost:8088/departments URL을 GET 방식으로 호출할 때 이 액션 메소드가 호출된다.

student 테이블의 레코드 전체 목록이 JSON 형식으로 출력된다.

http://localhost:8088/departments



JSON 형식의 학과 목록 데이터가 출력된다.



firefox 웹브라우저는 JSON 형식의 데이터를 보기 좋게 출력한다.

|  |
| --- |
| @GetMapping("department/{id}")  public Department department(@PathVariable("id") int id) {  return departmentRepository.findById(id).get();  } |

http://localhost:8088/department/1 URL을 GET 방식으로 호출할 때 이 액션 메소드가 호출된다.

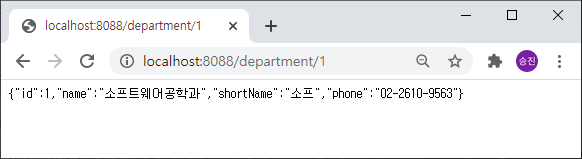
id 필드 값이 1인 학과 레코드가 JSON 형식으로 출력된다.

departmentRepository.findById(id)

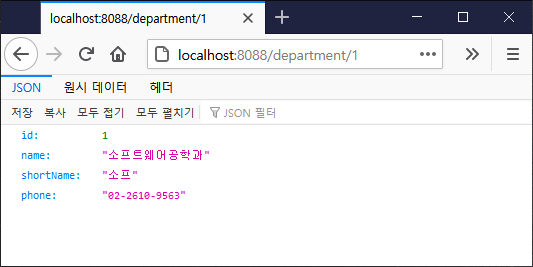
이 메소드의 리턴 값은 Optional<Department> 타입의 객체이다.

이 객체의 get 메소드를 호출하면 Department 객체가 리턴된다.

http://localhost:8088/department/1



JSON 형식의 학과 데이터가 출력된다.



firefox 웹브라우저는 JSON 형식의 데이터를 보기 좋게 출력한다.

# 부록 - frontend 앱

## departments.css

### src/main/webapp/departments.css

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | /\* 공통 \*/  button { padding: 0.4em 1.5em; }  table { border-collapse: collapse; margin-bottom: 20px; width: 100%; }  tr:nth-child(1) { background-color: #ddd; }  tr.clickable:hover { background-color: #ffb; cursor: pointer; }  td { padding: 8px; border: 1px solid #aaa; }  /\* main \*/  div#main { padding: 30px; margin: 30px auto; max-width: 800px;  border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa; }  div#main h1 { border-bottom: 1px solid gray; }  div#main button { float: right; margin-bottom: 5px; }  /\* dialog \*/  div#backdrop { position: absolute; left: 0; top: 0; width: 100%; height: 100%;  background-color: #aaa; opacity: 0.5; }  div#dialog { position: absolute; top: 50px; left: 50%;  padding: 30px; margin-left: -250px; max-width: 500px;  background-color: white;  border: 1px solid #ccc; box-shadow: 4px 4px 4px #666; }  div#dialog h1 { border-bottom: 1px solid gray; }  div#dialog input { padding: 5px; width: 100%; margin-bottom: 20px; font-size: 10pt; } |

## departments.html

### src/main/webapp/departments.html

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92 | <html lang="ko">  <head>  <meta charset="utf-8" />  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>  <script src="https://unpkg.com/axios/dist/axios.min.js"></script>  <link rel="stylesheet" type="text/css" href="departments.css">  </head>  <body>  <div id="app">  <div id="main">  <h1> 학과목록</h1>  <button type="button" v-on:click="create"> 추가</button>  <table>  <tr>  <td>id</td>  <td>학과명</td>  <td>단축명</td>  <td>전화</td>  </tr>  <tr v-for="department in departments" v-on:click="edit(department)"  v-bind:key="department.id" class="clickable">  <td>{{ department.id }}</td>  <td>{{ department.name }}</td>  <td>{{ department.shortName }}</td>  <td>{{ department.phone }}</td>  </tr>  </table>  </div>    <div id="backdrop" v-if="showDialog"></div>  <div id="dialog" v-if="showDialog">  <h1>학과정보 {{ showDialog }}</h1>  <label>학과명</label>  <input type="text" v-model.trim="department.name" placeholder="학과명"/>  <label>단축명</label>  <input type="text" v-model.trim="department.shortName" placeholder="단축명"/>  <label>전화번호</label>  <input type="text" v-model.trim="department.phone" placeholder="전화번호"/>  <div>  <button type="button" v-on:click="save"> 저장</button>  <button type="button" v-on:click="remove"> 삭제</button>  <button type="button" v-on:click="showDialog = false"> 취소</button>  </div>  </div>  </div>  <script type="text/javascript">  var app = new Vue({  el: '#app',  data: {  departments: [],  department: {},  showDialog: false  },  methods: {  create: function() {  this.showDialog = "등록"; ;  this.department = {};  },  reload: function() {  axios.get('departments')  .then(response => this.departments = response.data );  },  edit: function(department) {  this.showDialog = "수정";  this.department = Object.assign({}, department);  },  save: function() {  if (this.showDialog == "등록")  axios.post('department', this.department)  .then(response => this.reload());  else  axios.put('department', this.department)  .then(response => this.reload());  this.showDialog = false;  },  remove: function() {  if (confirm("삭제하시겠습니까?") == false) return;  axios.delete('department/' + this.department.id)  .then(response => this.reload());  this.showDialog = false;  }  }  })  window.onload = function() {  app.reload();  }  </script>  </body>  </html> |

지난 학기 프론트엔드개발에서 배웠던 todo 앱 구현과 동일한 방법으로 학과목록 앱을 구현함.

