**Lab03**

보고서

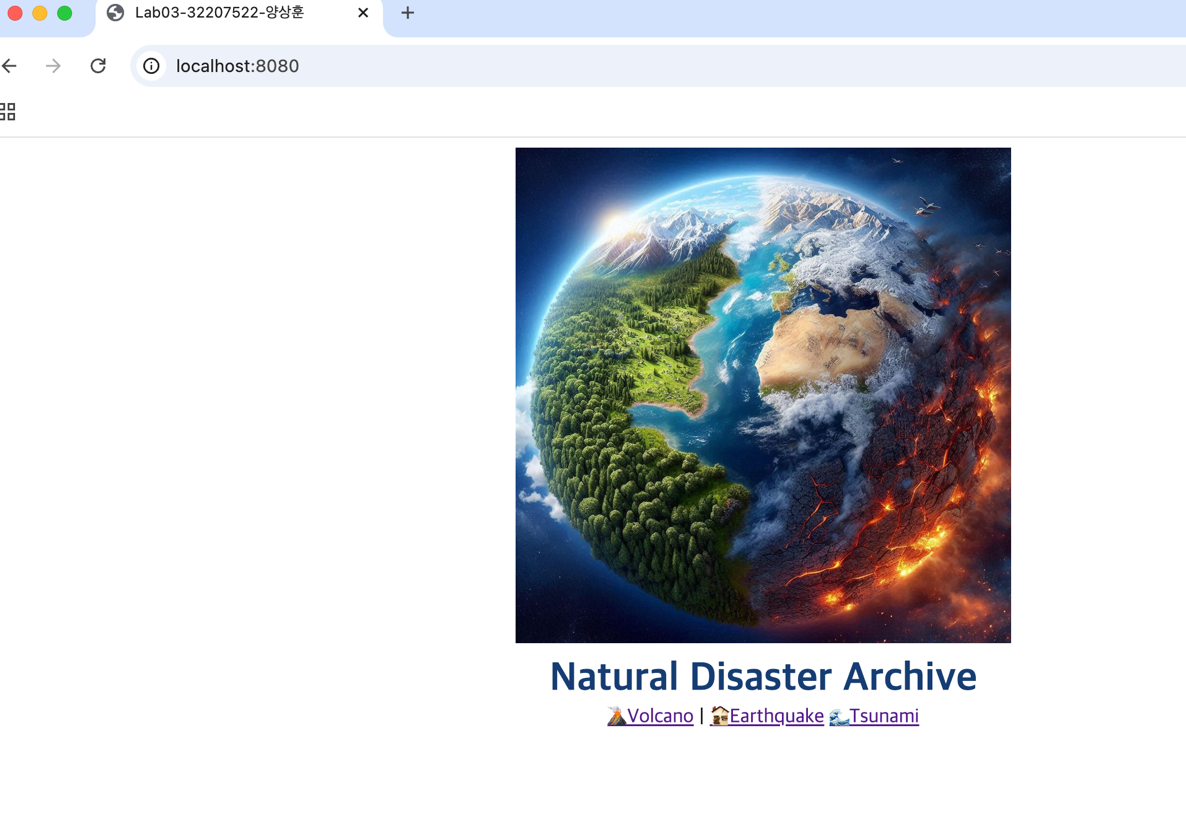
Lab03 문제해결 프로그래밍 설계 1분반

32207522 양상훈

제출일 : 25/04/15

**실행결과(Volcano 이미지만, Earthquake, Tsunami 모두 동일함.)**

**초기화면**



**Volcano 링크 클릭**

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**아무 입력 없이 Search 클릭**

텍스트, 번호, 소프트웨어, 스크린샷이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**Min Year = 2023 입력하고 Search 클릭**

**텍스트, 소프트웨어, 번호, 스크린샷이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.**

**소스 코드(코드 양이 많아서 Volcano에 대한 부분만 작성하겠습니다. Earthquake, Tsunami 구조는 동일합니다.)**

**컨트롤러**

package kr.ac.dankook.ace.lab3.controller;

import kr.ac.dankook.ace.lab3.dto.\*;

import kr.ac.dankook.ace.lab3.form.SearchForm;

import kr.ac.dankook.ace.lab3.service.\*;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RequiredArgsConstructor

@Controller

public class DisasterController {

private final VolcanoService volcanoService;

private final EarthquakeService earthquakeService;

private final TsunamiService tsunamiService;

@GetMapping("/")

public String home() {

return "lab3";

}

// Volcano 검색폼 페이지

@GetMapping("/volcano")

public String showVolcanoForm(Model model) {

SearchForm form = new SearchForm();

form.setDisasterType("volcano");

model.addAttribute("searchForm", form);

return "form"; // form.html

}

// Volcano 검색 결과 처리

@PostMapping("/search/volcano")

public String searchVolcano(@ModelAttribute("searchForm") SearchForm form, Model model) {

List<Volcano> results = volcanoService.search(form);

model.addAttribute("searchForm", form);

model.addAttribute("results", results);

return "form"; // 결과도 같은 페이지에 출력

}

// Earthquake 검색폼 페이지

@GetMapping("/earthquake")

public String showEarthquakeForm(Model model) {

SearchForm form = new SearchForm();

form.setDisasterType("earthquake");

model.addAttribute("searchForm", form);

return "form";

}

// Earthquake 검색 결과 처리

@PostMapping("/search/earthquake")

public String searchEarthquake(@ModelAttribute("searchForm") SearchForm form, Model model) {

List<Earthquake> results = earthquakeService.search(form);

model.addAttribute("searchForm", form);

model.addAttribute("results", results);

return "form";

}

// Tsunami 검색폼 페이지 -> dto 쓰나미의 필드 데이터를 form.html에 searchForm 이라는 이름으로 넘긴다.

@GetMapping("/tsunami")

public String showTsunamiForm(Model model) {

SearchForm form = new SearchForm();

form.setDisasterType("tsunami");

model.addAttribute("searchForm", form);

return "form"; // form.html

}

// Tsunami 검색 결과 처리

@PostMapping("/search/tsunami")

public String searchTsunami(@ModelAttribute("searchForm") SearchForm form, Model model) {

List<Tsunami> results = tsunamiService.search(form);

model.addAttribute("searchForm", form);

model.addAttribute("results", results);

return "form";

}

}

**서비스**

package kr.ac.dankook.ace.lab3.service;

import kr.ac.dankook.ace.lab3.dao.dataimporter.DataImporter;

import kr.ac.dankook.ace.lab3.dao.dataimporter.VolcanoDataImporter;

import kr.ac.dankook.ace.lab3.dao.finder.VolcanoFinder;

import kr.ac.dankook.ace.lab3.dao.parser.VolcanoCleanParser;

import kr.ac.dankook.ace.lab3.dao.parser.VolcanoRawParser;

import kr.ac.dankook.ace.lab3.dto.Volcano;

import kr.ac.dankook.ace.lab3.form.SearchForm;

import org.springframework.stereotype.Service;

import jakarta.annotation.PostConstruct;

import java.util.List;

@Service

public class VolcanoService {

// raw 파서를 이용해서 raw 데이터를 정제

private final DataImporter<Volcano> rawImporter = new VolcanoDataImporter(new VolcanoRawParser());

// clean 파서로 정제된 csv 데이터를 다시 로드

private final DataImporter<Volcano> cleanImporter = new VolcanoDataImporter(new VolcanoCleanParser());

private final VolcanoFinder finder = new VolcanoFinder();

private final String volcanoCsvPath = "src/main/resources/static/data/volcano.csv";

private final String cleanCsvPath = "src/main/resources/static/data/volcano\_clean.csv";

@PostConstruct

public void init() {

cleanVolcanoData();

}

public void cleanVolcanoData() {

List<Volcano> rawList = rawImporter.load(volcanoCsvPath);

cleanImporter.save(cleanCsvPath, rawList);

}

public List<Volcano> search(SearchForm form) {

List<Volcano> volcanoes = cleanImporter.load(cleanCsvPath);

volcanoes = finder.findByYearRange(volcanoes, form.getMinYear(), form.getMaxYear());

volcanoes = finder.findByVEIRange(volcanoes, form.getMinVei(), form.getMaxVei());

volcanoes = finder.findByCountry(volcanoes, form.getCountry());

volcanoes = finder.findByLocationContains(volcanoes, form.getLocation());

volcanoes = finder.findByNameContains(volcanoes, form.getName());

return volcanoes;

}

}

**DAO(Parser, Impoter, Finder)**

**Parser 인터페이스**

package kr.ac.dankook.ace.lab3.dao.parser;

/\*

\* RawParser : 기존 csv 파일을 파싱

\* CleanParser : 새로 만들어진 csv 파일을 파싱

\*/

public interface Parser<T> {

T parse(String line);

}

**VolcanoRawParser**

package kr.ac.dankook.ace.lab3.dao.parser;

import kr.ac.dankook.ace.lab3.dto.Volcano;

public class VolcanoRawParser implements Parser<Volcano> {

@Override

public Volcano parse(String line) {

String[] tokens = line.split(",", -1);

if (tokens.length < 14) return null;

Volcano volcano = new Volcano();

volcano.setYear(parseInt(tokens[1]));

volcano.setMonth(parseInt(tokens[2]));

volcano.setDay(parseInt(tokens[3]));

volcano.setName(tokens[6]);

volcano.setLocation(tokens[7]);

volcano.setCountry(tokens[8]);

volcano.setVei(parseInt(tokens[13]));

return volcano;

}

private int parseInt(String s) {

try { return Integer.parseInt(s.trim()); } catch (Exception e) { return -1; }

}

}

**VolcanoCleanParser**

package kr.ac.dankook.ace.lab3.dao.parser;

import kr.ac.dankook.ace.lab3.dto.Volcano;

public class VolcanoCleanParser implements Parser<Volcano> {

@Override

public Volcano parse(String line) {

String[] tokens = line.split(",");

if (tokens.length < 7) return null;

Volcano volcano = new Volcano();

volcano.setYear(parseInt(tokens[0]));

volcano.setMonth(parseInt(tokens[1]));

volcano.setDay(parseInt(tokens[2]));

volcano.setName(tokens[3]);

volcano.setLocation(tokens[4]);

volcano.setCountry(tokens[5]);

volcano.setVei(parseInt(tokens[6]));

return volcano;

}

private int parseInt(String s) {

try { return Integer.parseInt(s.trim()); } catch (Exception e) { return -1; }

}

}

**DataImporter 인터페이스**

package kr.ac.dankook.ace.lab3.dao.dataimporter;

import java.util.List;

public interface DataImporter<T> {

List<T> load(String filepath);

void save(String filepath, List<T> data);

}

**VolcanoDataImporter**

package kr.ac.dankook.ace.lab3.dao.dataimporter;

import kr.ac.dankook.ace.lab3.dao.parser.Parser;

import kr.ac.dankook.ace.lab3.dto.Volcano;

import java.io.\*;

import java.util.\*;

public class VolcanoDataImporter implements DataImporter<Volcano> {

private final Parser<Volcano> parser;

public VolcanoDataImporter(Parser<Volcano> parser) {

this.parser = parser;

}

@Override

//csv 파일 가져오기

public List<Volcano> load(String filepath) {

List<Volcano> list = new ArrayList<>();

try (BufferedReader br = new BufferedReader(new FileReader(filepath))) {

br.readLine(); // 첫 줄 스킵 (헤더)

String line;

while ((line = br.readLine()) != null) {

Volcano v = parser.parse(line);

if (v != null) list.add(v);

}

} catch (IOException e) {

e.printStackTrace();

}

return list;

}

@Override

// 정제된 CSV로 저장

public void save(String filepath, List<Volcano> volcanoes) {

try (BufferedWriter bw = new BufferedWriter(new FileWriter(filepath))) {

bw.write("Year,Month,Day,Name,Location,Country,Vei");

bw.newLine();

for (Volcano v : volcanoes) {

bw.write(String.join(",",

String.valueOf(v.getYear()),

String.valueOf(v.getMonth()),

String.valueOf(v.getDay()),

sanitize(v.getName()),

sanitize(v.getLocation()),

sanitize(v.getCountry()),

String.valueOf(v.getVei())

));

bw.newLine();

}

} catch (IOException e) {

e.printStackTrace();

}

}

// 문자열 정제 (null 방지 + 큰 따옴표 제거)

private String sanitize(String input) {

if (input == null) return "";

return input.replace("\"", "");

}

}

**VolcanoFinder**

package kr.ac.dankook.ace.lab3.dao.finder;

import kr.ac.dankook.ace.lab3.dto.Volcano;

import java.util.List;

import java.util.stream.Collectors;

// 사용자가 입력하지 않거나 null인 경우 넘어감.

public class VolcanoFinder {

public List<Volcano> findByYearRange(List<Volcano> volcanoes, Integer minYear, Integer maxYear) {

if (minYear == null && maxYear == null) return volcanoes;

return volcanoes.stream()

.filter(v -> (minYear == null || v.getYear() >= minYear))

.filter(v -> (maxYear == null || v.getYear() <= maxYear))

.collect(Collectors.toList());

}

public List<Volcano> findByVEIRange(List<Volcano> volcanoes, Integer minVei, Integer maxVei) {

if (minVei == null && maxVei == null) return volcanoes;

return volcanoes.stream()

.filter(v -> (minVei == null || v.getVei() >= minVei))

.filter(v -> (maxVei == null || v.getVei() <= maxVei))

.collect(Collectors.toList());

}

public List<Volcano> findByCountry(List<Volcano> volcanoes, String country) {

if (country == null || country.isBlank()) return volcanoes;

return volcanoes.stream()

.filter(v -> v.getCountry().toLowerCase().contains(country.toLowerCase()))

.collect(Collectors.toList());

}

public List<Volcano> findByLocationContains(List<Volcano> volcanoes, String location) {

if (location == null || location.isBlank()) return volcanoes;

return volcanoes.stream()

.filter(v -> v.getLocation().toLowerCase().contains(location.toLowerCase()))

.collect(Collectors.toList());

}

public List<Volcano> findByNameContains(List<Volcano> volcanoes, String name) {

if (name == null || name.isBlank()) return volcanoes;

return volcanoes.stream()

.filter(v -> v.getName().toLowerCase().contains(name.toLowerCase()))

.collect(Collectors.toList());

}

}

**DTO**

package kr.ac.dankook.ace.lab3.dto;

import lombok.Getter;

import lombok.Setter;

// Lombok 사용하여 getters and setters 생략

@Getter @Setter

public class Volcano {

private int year;

private int month;

private int day;

private String name;

private String location;

private String country;

private int vei;

// 정확한 날짜 출력을 위함.

public String getDate() {

return String.format("%04d-%02d-%02d", year, month, day);

}

}

**SearchForm**

package kr.ac.dankook.ace.lab3.form;

import lombok.Getter;

import lombok.Setter;

@Getter @Setter

public class SearchForm {

private String disasterType; // volcano, earthquake, tsunami 구분

// 공통 필드

private Integer minYear;

private Integer maxYear;

// volcano, tsunami 공통 필드

private String country;

private String location;

private String name;

// earthquake, tsunami 공통 필드

private String locationName;

// volcano 전용 필드

private Integer minVei;

private Integer maxVei;

// earthquake 전용 필드

private Double minMagnitude;

private Double maxMagnitude;

private Double maxDepth;

// tsunami 전용 필드

private Double maximumWaterHeight;

}

**코드 설명**

csv 파일이 복잡하기 때문에 정제된 csv 파일을 만들었음.

1. RawParser로 복잡한 csv 파일에서 필요한 정보만을(DTO 멤버 필드) 파싱

2. DataImporter를 통해 csv 파일을 load하고 clean한 csv 파일을 save

3. CleanParser로 정제된 csv 파일을 파싱하도록 함.

4. SearchForm에는 form.html에서의 조건 검색에 필요한 멤버 필드들을 생성

5. Finder는 조건 검색을 만족할 수 있도록 하는 기능 (ex. minYear >= 2023 -> 검색 폼에서 Year가 2023 이상인 정보들만 보이도록 함.) -> 필터링 기능

6. 이러한 로직들은 Service에서 처리된다.

**코드 실행 흐름**

1.사용자 요청(화면에서 필터링 조건을 입력하고 Search를 클릭)

2.SearchForm에서 Request DTO

3.Service 로직 실행

4.DAO(DataImporter, Finder, Parser) : 데이터에 접근

5.데이터 조회 결과

6.Response DTO

7.사용자에게 응답(테이블로 필터링된 데이터를 보여줌)