Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №6

по дисциплине «Программирование на языках высокого уровня»

«Concurrency»

Выполнил: Проверил:

Снитко Д.А. ассистент Скиба И.Г.

гр.250501

Минск 2024

**1. Постановка задачи**

1. Добавить сервис для подсчёта обращений к основному сервису. Счётчик должен быть реализован в виде отдельного класса, доступ к которому должен быть синхронизирован.

2. Используя jmeter/postman или любые другие средства сконфигурировать нагрузочный тест и убедиться, что счётчик обращений работает правильно при большой нагрузке.

**2. Структура проекта**

В проекте используется послойная архитектура из нескольких пакетов,

которые отвечают за определенные функции.

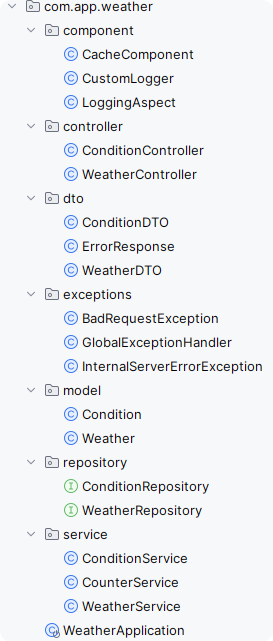


Рисунок 2.1 – Структура проекта

**3. Листинг кода**

Файл ConditionController.java

package com.app.weather.controller;

import com.app.weather.component.CustomLogger;

import com.app.weather.dto.ConditionDTO;

import com.app.weather.model.Condition;

import com.app.weather.service.ConditionService;

import com.app.weather.service.CounterService;

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/conditions")

public class ConditionController {

private static final String COUNTER\_MSG = "\nCounter: ";

private final ConditionService conditionService;

private final CounterService counterService;

private final CustomLogger customLogger;

public ConditionController(ConditionService conditionService, CustomLogger customLogger, CounterService counterService) {

this.conditionService = conditionService;

this.customLogger = customLogger;

this.counterService = counterService;

}

@PostMapping("/bulk")

public ResponseEntity<List<ConditionDTO>> createConditionBulk(@RequestBody List<ConditionDTO> conditionDTOs) {

customLogger.info("Creating bulk of conditions" + COUNTER\_MSG + counterService.incrementAndGet());

List<Condition> createdConditions = conditionService.createConditionBulk(conditionDTOs);

return ResponseEntity.ok(createdConditions.stream()

.map(conditionService::convertToDTO)

.toList());

}

@GetMapping

public ResponseEntity<List<ConditionDTO>> getAllConditions() {

customLogger.info("Получение всех condition" + COUNTER\_MSG + counterService.incrementAndGet());

List<Condition> conditions = conditionService.getAllConditions();

List<ConditionDTO> conditionDTOs = conditions.stream()

.map(conditionService::convertToDTO)

.toList();

return ResponseEntity.ok(conditionDTOs);

}

@GetMapping("/{id}")

public ResponseEntity<ConditionDTO> getConditionById(@PathVariable Long id) {

customLogger.info("Получение condition по id: " + id + COUNTER\_MSG + counterService.incrementAndGet());

Condition condition = conditionService.getConditionById(id);

if (condition == null) {

return ResponseEntity.notFound().build();

}

return ResponseEntity.ok(conditionService.convertToDTO(condition));

}

@PostMapping

public ResponseEntity<ConditionDTO> createCondition(@RequestBody ConditionDTO conditionDTO) {

customLogger.info("Создание условия" + COUNTER\_MSG + counterService.incrementAndGet());

Condition condition = conditionService.convertToEntity(conditionDTO);

Condition savedCondition = conditionService.createCondition(condition);

return ResponseEntity.ok(conditionService.convertToDTO(savedCondition));

}

@PutMapping("/{id}")

public ResponseEntity<ConditionDTO> updateCondition(@PathVariable Long id, @RequestBody ConditionDTO conditionDTO) {

customLogger.info("Обновление condition с id: " + id + COUNTER\_MSG + counterService.incrementAndGet());

Condition updatedCondition = conditionService.updateCondition(id, conditionDTO);

if (updatedCondition == null) {

return ResponseEntity.notFound().build();

}

return ResponseEntity.ok(conditionService.convertToDTO(updatedCondition));

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteCondition(@PathVariable Long id) {

customLogger.info("Удаление condition с id: " + id + COUNTER\_MSG + counterService.incrementAndGet());

if (!conditionService.deleteCondition(id)) {

return ResponseEntity.notFound().build();

}

return ResponseEntity.noContent().build();

}

}

Файл WeatherController.java

package com.app.weather.controller;

import com.app.weather.component.CustomLogger;

import com.app.weather.dto.WeatherDTO;

import com.app.weather.model.Weather;

import com.app.weather.service.CounterService;

import com.app.weather.service.WeatherService;

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/weather")

public class WeatherController {

private static final String COUNTER\_MSG = "\nCounter: ";

private final WeatherService weatherService;

private final CounterService counterService;

private final CustomLogger customLogger;

public WeatherController(WeatherService weatherService, CustomLogger customLogger, CounterService counterService) {

this.weatherService = weatherService;

this.customLogger = customLogger;

this.counterService = counterService;

}

@PostMapping("/bulk")

public ResponseEntity<List<WeatherDTO>> createWeatherBulk(@RequestBody List<WeatherDTO> weatherDTOs) {

customLogger.info("Creating bulk of weathers" + COUNTER\_MSG + counterService.incrementAndGet());

List<Weather> createdWeathers = weatherService.createWeatherBulk(weatherDTOs);

return ResponseEntity.ok(createdWeathers.stream()

.map(weatherService::convertToDTO)

.toList());

}

@GetMapping

public ResponseEntity<List<WeatherDTO>> getAllWeathers() {

customLogger.info("Получение всех weather" + COUNTER\_MSG + counterService.incrementAndGet());

List<Weather> weathers = weatherService.getAllWeathers();

List<WeatherDTO> weatherDTOs = weathers.stream()

.map(weatherService::convertToDTO)

.toList();

return ResponseEntity.ok(weatherDTOs);

}

@GetMapping("/{id}")

public ResponseEntity<WeatherDTO> getWeatherById(@PathVariable Long id) {

customLogger.info("Получение weather по id: " + id + COUNTER\_MSG + counterService.incrementAndGet());

Weather weather = weatherService.getWeatherById(id);

if (weather == null) {

return ResponseEntity.notFound().build();

}

return ResponseEntity.ok(weatherService.convertToDTO(weather));

}

@GetMapping("/citiesT/{temperature}")

public ResponseEntity<List<WeatherDTO>> getWeatherByTemperature(@PathVariable double temperature) {

customLogger.info("Получение weather по температуре: " + temperature + COUNTER\_MSG + counterService.incrementAndGet());

List<WeatherDTO> weatherDTOs = weatherService.findByTemperature(temperature);

return ResponseEntity.ok(weatherDTOs);

}

@GetMapping("/citiesC/{conditionText}")

public ResponseEntity<List<WeatherDTO>> findByConditionText(@PathVariable String conditionText) {

customLogger.info("Получение condition по тексту условия: " + conditionText + COUNTER\_MSG + counterService.incrementAndGet());

List<WeatherDTO> weathers = weatherService.findByConditionText(conditionText);

return ResponseEntity.ok(weathers);

}

@PostMapping

public ResponseEntity<WeatherDTO> createWeatherWithConditionText(@RequestBody WeatherDTO weatherDTO) {

customLogger.info("Создание weather с condition" + COUNTER\_MSG + counterService.incrementAndGet());

Weather createdWeather = weatherService.createWeatherWithCondition(weatherDTO);

return ResponseEntity.ok(weatherService.convertToDTO(createdWeather));

}

@PutMapping("/{id}")

public ResponseEntity<WeatherDTO> updateWeather(@PathVariable Long id, @RequestBody WeatherDTO weatherDTO) {

customLogger.info("Обновление weather с id: " + id + COUNTER\_MSG + counterService.incrementAndGet());

Weather weather = weatherService.updateWeather(id, weatherDTO);

if (weather == null) {

return ResponseEntity.notFound().build();

}

return ResponseEntity.ok(weatherService.convertToDTO(weather));

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteWeather(@PathVariable Long id) {

customLogger.info("Удаление weather с id: " + id + COUNTER\_MSG + counterService.incrementAndGet());

weatherService.deleteWeather(id);

return ResponseEntity.noContent().build();

}

}

Файл CounterService.java

package com.app.weather.service;

import org.springframework.stereotype.Service;

import java.util.concurrent.atomic.AtomicInteger;

@Service

public class CounterService {

private final AtomicInteger count;

public CounterService() {

count = new AtomicInteger(0);

}

public int incrementAndGet() {

return count.incrementAndGet();

}

public int get() {

return count.get();

}

}

**4. Результат программы**

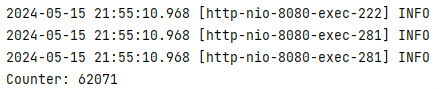


Рисунок 4.1 – Работа счетчика

**5. Заключение**

В результате работы был добавлен сервис для подсчета обращений к основному сервису. Счетчик был реализован в виде отдельного класса с использованием AtomicInteger. Также был проведен нагрузочный тест с использованием средства JMeter, который подтвердил корректную работу счетчика обращений при большой нагрузке.