МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

АНАЛИЗ И СРАВНЕНИЕ ФРЕЙМВОРКОВ SPRING MVC И SPRING WEBFLUX НА ПРИМЕРЕ РЕАЛИЗАЦИИ ИНТЕРНЕТ-МАГАЗИНА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы
направления 02.03.03 Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Путятинского Дмитрия Евгеньевича

Научный руководитель:
Старший преподаватель _____ А.А. Казачкова

подпись, дата

Зав. кафедрой:
к.ф.-м.н., доцент _____ М.В. Огнева

подпись, дата

Саратов 2022

ВВЕДЕНИЕ

Актуальность темы.

Постепенно с развитием интернета стали появляться высоконагруженные системы, с которыми работает большое количество пользователей одновременно. Примером такой системы является интернет-магазин, с которым работает множество клиентов. Если большое количество запросов перегрузят систему, то работа приложения замедлится, пользователям придется долго ждать отклика приложения, что в конечном счете приведет к потере клиентов, которые отдадут предпочтение более быстрому приложению. Одним из способов устранения данной проблемы является реактивное программирование.

Реактивное программирование — парадигма программирования, ориентированная на потоки данных и распространение изменений. Это означает, что должна существовать возможность легко выражать статические и динамические потоки данных, а также то, что нижележащая модель исполнения должна автоматически распространять изменения благодаря потоку данных.

Однако реактивное программирование подходит не во всех случаях и иногда лучше остановится на привычном способе обработки запросов. Необходимо понимать в каких случаях реактивное программирование действительно необходимо и может ускорить работу приложения, а в каких лучше отдать предпочтение стандартному подходу.

Во многих языках программирования существуют свои реализации реактивного подхода. В языке Java реактивный подход реализован в микрофреймворке Spring WebFlux.

Модуль WebFlux появился в пятой версии фреймворка Spring. Этот микрофреймворк является дополнением Spring MVC и отражает собой реактивный подход для написания веб-сервисов.

Цель бакалаврской работы – анализ и сравнение фреймворков Spring MVC и Spring WebFlux на примере реализации интернет-магазина.

Поставленная цель определила **следующие задачи**:

- Изучить клиент-серверную архитектуру и основные понятия;

- Изучить особенности фреймворка Spring MVC;
- Изучить особенности фреймворка Spring WebFlux;
- Сравнить между собой подходы, которые лежат в основе данных фреймворков;
- Реализовать приложение с использованием Spring MVC;
- Реализовать приложение с добавлением Spring Webflux;
- Сравнить производительность реализованных приложениях при различной нагрузке.

Методологические основы по реализации веб-приложений на фреймворках Spring WebFlux и Spring MVC представлены в работах Крейга Уоллса, Игоря Лозинского, Олега Докуки и Эндрю Ломбарди и Джозефа Оттингера.

Теоретическая значимость бакалаврской работы.

Теоретическая значимость данной бакалаврской работы состоит в изучении особенностей реактивного программирования, и изучении архитектурных и реализационных особенностей фреймворков Spring MVC и Spring WebFlux.

Практическая значимость бакалаврской работы.

Практическая значимость данной бакалаврской работы состоит в том, что на примере, реализованных в ходе работы, приложений можно сделать выводы о преимуществах реактивного подхода в высоконагруженных системах, относительно привычного подхода с блокирующим стеком.

Структура и объем работы.

Бакалаврская работа состоит из введения, шести разделов, заключения, списка использованных источников и шести приложений. Общий объем работы – 69 страниц, из них 55 страниц – основное содержание, включая 9 рисунков и 1 таблицу, цифровой носитель в качестве приложения, список использованных источников информации – 20 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Клиент-серверная архитектура и основные понятия» посвящен основным понятиям, связанным с построением веб-приложений с клиент-серверной архитектурой.

1.1 HTTP-протокол

В данном подразделе описывается HTTP-протокол, его появление и применение.

1.2 Архитектурный стиль REST

В данном подразделе рассмотрен архитектурный стиль REST, его ограничения, свойства и преимущества REST-приложений.

1.3 Трехуровневая архитектура

В данном подразделе рассмотрена трехуровневая архитектура, ее особенности и преимущества относительно двухзвенной клиент-серверной архитектурой.

1.4 Реактивное программирование

В данном подразделе рассмотрено реактивное программирования и его преимущества.

Второй раздел «Особенности фреймворка Spring MVC» посвящен архитектуре фреймворка Spring MVC и деталям реализации веб-приложений с использованием данного фреймворка.

2.1 Функции фреймворка Spring MVC

В данном подразделе описаны уникальные функции веб-поддержки, которыми обладает Spring MVC.

2.2 Роль DispatcherServlet в Spring MVC

В данном подразделе описывается принцип работы фреймворка, а также роль Dispatcher Servlet и других компонентов фреймворка в архитектуре веб-приложения.

2.3 Асинхронная обработка запросов в Spring MVC

В данном подразделе рассматривается возможность асинхронной обработки запросов в Spring MVC, ее особенности и принципы работы.

Третий раздел «Особенности фреймворка Spring WebFlux» посвящен архитектуре фреймворка Spring WebFlux и деталям реализации веб-приложений с использованием данного фреймворка.

3.1 Реактивное программирование

В данном подразделе более подробно раскрывается понятие реактивного программирования и описывается модель цикла событий (модель реактивного асинхронного программирования для серверов).

3.2 Реактивный фреймворк Spring WebFlux

В данном подразделе рассмотрены причины появления данного фреймворка.

3.3 Реактивный интерфейс Reactive API

В данном подразделе рассмотрены: реактивный интерфейс Reactive API, играющий важную роль в обеспечении функциональной совместимости; реактивная библиотека Reactor, которая предоставляет API-интерфейсы Mono и Flux для работы с последовательностями данных; спецификация Reactive Streams, которая определяет взаимодействие между асинхронными компонентами с противодавлением.

3.4 Поддержка серверов

В данном подразделе рассмотрены сервера, которые поддерживает Spring WebFlux, и особенности при работе с этими серверами.

3.5 Преимущество реактивного подхода

В данном подразделе рассмотрены преимущества реактивного подхода, модель параллелизма и модель потоков, которые используются в Spring WebFlux.

3.6 Модель параллелизма

В данном подразделе рассмотрена модель параллелизма, которая применяется в приложениях, реализованных на фреймворке Spring WebFlux.

3.7 Модель потоков

В данном подразделе рассмотрен принцип создания потоков в веб-приложениях, реализованных на фреймворке Spring WebFlux.

Четвертый раздел «Сравнение фреймворков Spring MVC и Spring WebFlux» сравнивает функциональные возможности фреймворков. В данном разделе описаны возможности, которые поддерживает каждый из фреймворков. Что могут использовать оба фреймворка, а что каждый из них поддерживает уникальным образом. Также в данном разделе рассмотрены ситуации, когда лучше использовать фреймворк Spring MVC, а когда — Spring WebFlux.

Пятый раздел «Используемые библиотеки и технологии» посвящен обзору технологий и библиотек, которые использовались при реализации веб-приложений.

5.1 Среда разработки

В данном подразделе объясняется, почему для реализации приложений была выбрана среда разработки Intellij Idea, описываются ее преимущества и особенности

5.2 Автоматическая сборка проекта

В данном подразделе рассматривается фреймворк Apache Maven, его особенности, преимущества, жизненный цикл и управление зависимостями.

5.3 Используемые библиотеки

В данном подразделе рассматриваются библиотеки, которые использовались при реализации приложений. Здесь описывается библиотека Lombok и ее применение, а также библиотека логирования log4j и уровни логирования в данной библиотеке.

Шестой раздел «Использование фреймворков Spring MVC и Spring WebFlux на примере интернет магазина» посвящен реализации приложений интернет-магазинов на фреймворках Spring MVC и Spring WebFlux, а также сравнению скорости обработки запросов у получившихся приложений.

6.1 Формальная постановка задачи

В данном подразделе описана архитектура приложения интернет-магазина и его функционал. Подробно описаны возможности, каждой из трех подсистем веб-приложения, HTTP-запросы, которые можно отправить к приложению, принцип авторизации пользователя и возможности пользователя.

6.2 Реализация приложения с использованием Spring MVC

В данном подразделе описаны детали реализации приложения интернет-магазина на фреймворке Spring MVC. Описан принцип создания контроллеров, репозиториев, также описана настройка файла конфигурации и настройка модуля Spring Security, для аутентификации и авторизации пользователя в системе интернет-магазина.

6.3 Реализация приложения с использованием Spring WebFlux

В данном подразделе описаны детали реализации приложения интернет-магазина на фреймворке Spring WebFlux. Описаны отличия в реализации, относительно приложение, созданного на фреймворке Spring MVC. Рассмотрен R2dbc драйвер для взаимодействия с базой данных, рассмотрены особенности создания репозиториев и контроллеров.

6.4 Сравнение и анализ производительности приложений на Spring MVC и Spring WebFlux

Данный подраздел посвящен сравнению скорости обработки запросов у получившихся приложений. Сначала приводится сравнение производительности приложений на единичных запросах, потом производится сравнение при нагрузке в 1000 параллельных запросов. В данном разделе приведено время выполнения запросов каждым приложением и, на основе полученных данных, сделаны выводы о производительности фреймворков в зависимости от степени нагрузки на систему.

Итоги

По результатам тестирования можно сделать вывод, что при невысокой нагрузке приложение, построенное на фреймворке Spring MVC показывает себя лучше, чем аналогичное приложение, реализованное с использованием фреймворка Spring WebFlux. Это связано в первую очередь с затратам на асинхронную обработку запросов.

При нагрузочном тестировании с большим количеством запросов лучше показал себя фреймворк Spring WebFlux. Приложение, построенное на фреймворке Spring MVC, использует большое количество потоков, которые

отнимают друг у друга процессорное время, вследствии чего время выполнения запросов увеличивается. Кроме того, модель контейнера сервлетов Tomcat, который используется в приложении, основана на блокирующих вызовах. То есть, когда поток обращается к базе данных, поток блокируется и, пока не будет получен ответ, так и будет находится в заблокированном состоянии.

Из этого следует вывод, что в приложениях с большой нагрузкой целесообразно использовать реактивный стек, который будет быстрее обрабатывать запросы. В приложениях с небольшой нагрузкой лучше использовать стандартный подход с блокирующими вызовами.

Стоит отметить, что данное сравнение проводилось с настройками базы данных и приложения по умолчанию. Это сделано, чтобы показать различия между двумя различными подходами. В действительности при проектировании системы используются различные инструменты, которые могут оптимизировать работу приложения. К таким инструментам можно отнести индексы в используемой приложением базе данных, настройка серверов под конкретные нужды, с учетом предполагаемого количества пользователей, кэширование некоторых запросов пользователя и тд.

ЗАКЛЮЧЕНИЕ

В результате выполнения работы были решены поставленные задачи, и достигнута основная цель. Были реализованы интернет-магазины на фреймворках Spring MVC и Spring WebFlux, а также было проведено сравнение производительности данных приложений в нормальных условиях, и при высокой нагрузке. Были изучены как архитектурные, так и реализационные особенности в подходах, которые предоставляют данные фреймворки.

Также в рамках работы была изучена клиент-серверная архитектура, трехслойная архитектура, основы http-протокола и архитектурный стиль REST.

Были изучены детали реализации приложений на фреймворке Spring MVC, архитектура данного фреймворка и основные понятия и компоненты.

В ходе дальнейшей работы было произведено знакомство с основами реактивного программирования, и особенностями реализации приложений на основе фреймворка Spring WebFlux.

Также в ходе работы была использована библиотека Lombok, которая добавляет дополнительную функциональности в Java с помощью изменения исходного кода перед Java компиляцией.

Для отправки запросов использовались HTTP-клиент для тестирования API — Postman инструмент, используемый для тестирования загрузки веб-приложений — Apache JMeter.

По результатам работы можно сделать вывод, что при невысокой нагрузке приложение, построенное на фреймворке Spring MVC показывает себя лучше, чем аналогичное приложение, реализованное с использованием фреймворка Spring WebFlux. Но в высоконагруженных системах, с большим количеством запросов к серверу, лучше показал себя фреймворк Spring WebFlux благодаря особенностям реализации, а именно реактивному подходу.

Основные источники информации:

1. Web MVC framework: «Официальная документация» [Электронный ресурс]. URL:

https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html (дата обращения: 05.02.2022).

2. Крейг Уоллс. Spring in action — 6-е изд. — М.: «ДМК Пресс», 2022. — С. 752.
3. Web on Reactive Stack: «Официальная документация» [Электронный ресурс]. URL: https://docs.spring.io/spring-framework/docs/current/reference/html/web-reactive.html#webflux (дата обращения: 21.02.2022).
4. Joseph B. Ottinger, Andrew Lombardi. Beginning Spring 5: From Novice to Professional — 1-е изд. — М.: « Apress», 2020. — С. 379.
5. Baeldung: Concurrency in Spring WebFlux [Электронный ресурс]. URL: https://www.baeldung.com/spring-webflux-concurrency (дата обращения: 03.03.2022).
6. Baeldung: Backpressure Mechanism in Spring WebFlux [Электронный ресурс]. URL: https://www.baeldung.com/spring-webflux-backpressure (дата обращения: 07.03.2022).
7. Лозинский И., Докука Олег. Практика реактивного программирования в Spring 5 — 1-е изд. — М.: « ДМК Пресс», 2020. — С. 508.