Research on Multithreading in Spring Boot

Kyrgyz: Spring Boottогу көп жүйөлүүлүктүн принциптерин изилдөө **Russian:** Исследование принципов работы с многопоточностью в Spring Boot **English:** Research on Multithreading Principles in Spring Boot

Author Information

Name: Erlan Omoshov

Student of Department
Computer Science and Engineering,
Ala-Too International University
Email: erlan.omoshov@alatoo.edu.kg

Abstract

Kyrgyz: Бул макалада Spring Boottогу көп жүйөлүүлүктү ишке ашыруунун негизги принциптери каралат. Анда көп процессордук стратегиялар, анын ичинде ExecutorService жана @Async колдонуу жөнүндө сөз болот. Мунун натыйжасында, тиркемелердин натыйжалуулугу жогорулап, жооп берүү убактысы кыскартылат.

Russian: В статье рассматриваются ключевые принципы работы с многопоточностью в Spring Boot. Обсуждаются стратегии многопоточности, такие как использование ExecutorService и аннотации @Async, что позволяет оптимизировать производительность приложений и сократить время отклика.

English: This article examines the key principles of multithreading in Spring Boot. It discusses multithreading strategies, including the use of ExecutorService and the @Async annotation, which optimize application performance and reduce response time.

Keywords

Kyrgyz: Spring Boot, көп жүйөлүүлүк, @Async, ExecutorService, аткарууну оптималдаштыруу
Russian: Spring Boot, многопоточность, @Async, ExecutorService, оптимизация производительности
English: Spring Boot, multithreading, @Async, ExecutorService, performance optimization

---

Introduction

In the era of modern application development, multithreading is an indispensable tool for improving performance and responsiveness. Spring Boot simplifies multithreading implementation with built-in tools and configurations. This research focuses on understanding and applying core principles such as ExecutorService and @Async annotation for effective multithreading.

Theoretical Part

ExecutorService

ExecutorService provides an interface for managing thread pools and executing asynchronous tasks. Spring Boot facilitates its configuration via TaskExecutor for scalable and flexible thread management.

@Async Annotation

The @Async annotation enables asynchronous execution of methods. Combined with the @EnableAsync configuration, it simplifies the implementation of non-blocking operations.

Methodology

Research Objectives

1. To explore the use of ExecutorService and @Async in Spring Boot.
2. To evaluate multithreading performance through practical examples.

Methods Used

1. **Literature Review:** Analysis of existing frameworks and documentation.
2. **Experimental Implementation:** Testing multithreading with practical examples.

Practical Implementation

Example with ExecutorService:

```
@Service
public class AsyncService {
    @Async
    public void executeTask() {
        System.out.println("Running task in thread: " +
Thread.currentThread().getName());
    } }
```

Enabling Asynchronous Execution:

```
@Configuration
@EnableAsync
public class AsyncConfig {
    @Bean
    public Executor taskExecutor() {
        ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor();
        executor.setCorePoolSize(5);
        executor.setMaxPoolSize(10);
        executor.setQueueCapacity(25);
        executor.setThreadNamePrefix("Async-");
        executor.initialize();
        return executor;
    }
}
```

Conclusion

Multithreading is essential for enhancing the performance of Spring Boot applications. By leveraging tools such as ExecutorService and the @Async annotation, developers can build highly responsive and scalable systems. However, careful configuration and management of threads are crucial to ensure efficiency and avoid resource contention.

References

1. Spring Framework Documentation. URL: https://spring.io/projects/spring-framework
2. Java Concurrency in Practice. Author: Brian Goetz et al.

3. Spring Boot Reference Guide. URL: https://docs.spring.io/spring-boot/docs/current/reference/html/
4. Baeldung Tutorials on Spring Boot. URL: https://www.baeldung.com/