**Implementing Logging with Spring AOP**

This document explains how to implement logging with Spring AOP (Aspect-Oriented Programming) to track method execution times in a library management application. AOP allows you to add functionalities like logging without modifying the core business logic.

**Understanding the Requirement**

* We need to track how long methods in the BookService class take to execute.
* Logging these execution times can help identify performance bottlenecks in the application.

**Adding Spring AOP Dependency (Step 1)**

1. Update pom.xml to include the spring-aop dependency:
   * This dependency provides Spring AOP functionalities.

**Creating an Aspect for Logging (Step 2)**

1. Create a package com.library.aspect and add a class LoggingAspect:
   * This class will be our aspect responsible for logging.
2. Annotate LoggingAspect with @Aspect and @Component:
   * @Aspect marks the class as an Aspect.
   * @Component tells Spring to manage this class as a bean.
3. Define a method logExecutionTime annotated with @Around:
   * @Around indicates this advice will be invoked before and after the target method execution.
   * The method receives a ProceedingJoinPoint argument, providing information about the intercepted method.
4. Implement the logic in logExecutionTime:
   * Capture the start time using System.currentTimeMillis().
   * Call proceed.proceed() to execute the target method.
   * Calculate the execution time by subtracting the start time from the current time.
   * Log a message with the method signature and execution time.

**Enabling AspectJ Support (Step 3)**

1. Update applicationContext.xml to enable AspectJ and register the aspect:
   * Add the aop namespace declaration to reference Spring AOP schema elements.
   * Include an <aop:aspectj-autoproxy/> element to enable Spring AOP auto-proxying.
   * Define a bean for LoggingAspect using the <bean> element.

**Note:** The provided code snippet in the exercise solution has a corrected namespace declaration for AOP elements.

**Testing the Aspect (Step 4)**

1. Run the MainApp class:
   * Spring will create instances of BookService and LoggingAspect.
   * When bookService.manageBooks() is called, the LoggingAspect will intercept the call and log the execution time.
2. Observe the console output:
   * You should see a log message indicating the execution time of the manageBooks method in the BookService.

**Benefits of AOP Logging**

* Improved code modularity: Logging concerns are separated from business logic.
* Easier maintenance: Changes to logging configuration are centralized.
* Flexibility: Developers can define different aspects for various logging needs.

By utilizing Spring AOP, developers can enhance your application's functionality without modifying the core code. This exercise demonstrates a practical use case for AOP in logging method execution times.