AOP Tutorial

Written By: Muhammad Asif.

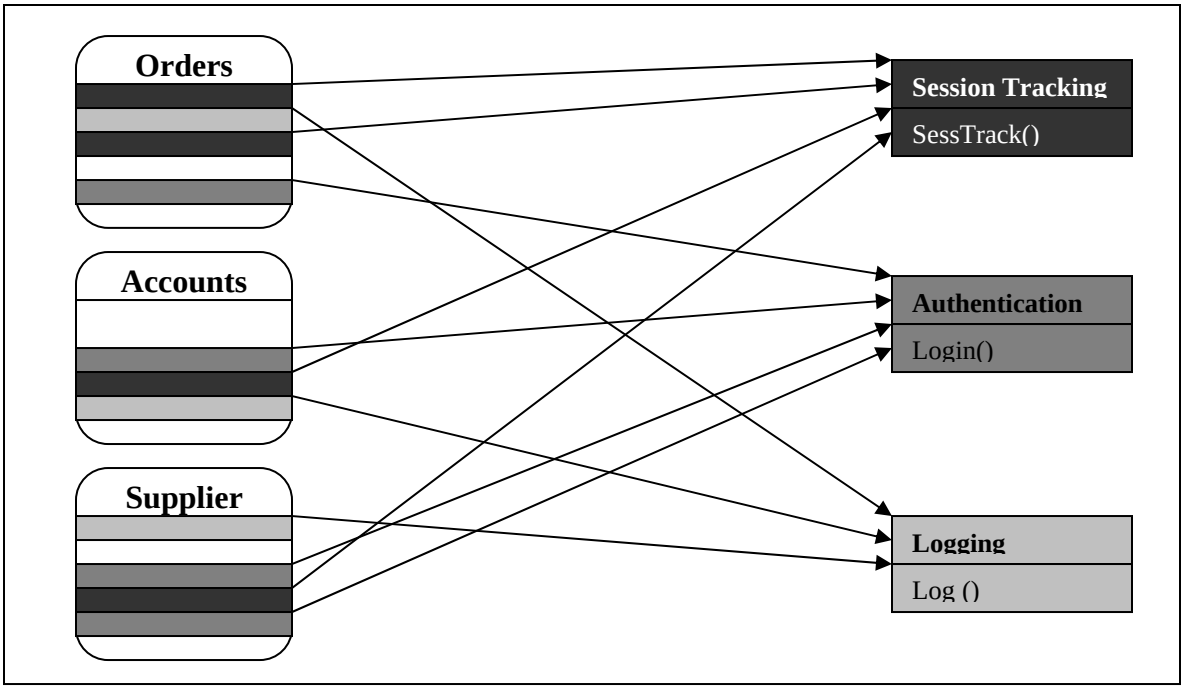
Department of Computer Science,
Virtual University of Pakistan

Table of Contents

1.0 Introduction

Software development is the most important field of computer science, it is the process of writing and maintaining the source code it includes tasks that covers conceptions of the desired software through to the final manifestation of the software, in order to achieve reliable, secure, and feasible results in given intervals of time a software development model is to be followed.

Software development model is a series of steps that are to be followed for a successful completion of a software project, there are a lot of software development models such as Waterfall Model (the most fundamental), Incremental Model and many more, but still there is no perfection in a single model; therefore we always want to introduce new techniques and methodologies.



s

Figure 1.0 Before AOP, the crosscutting code is injected in the original code

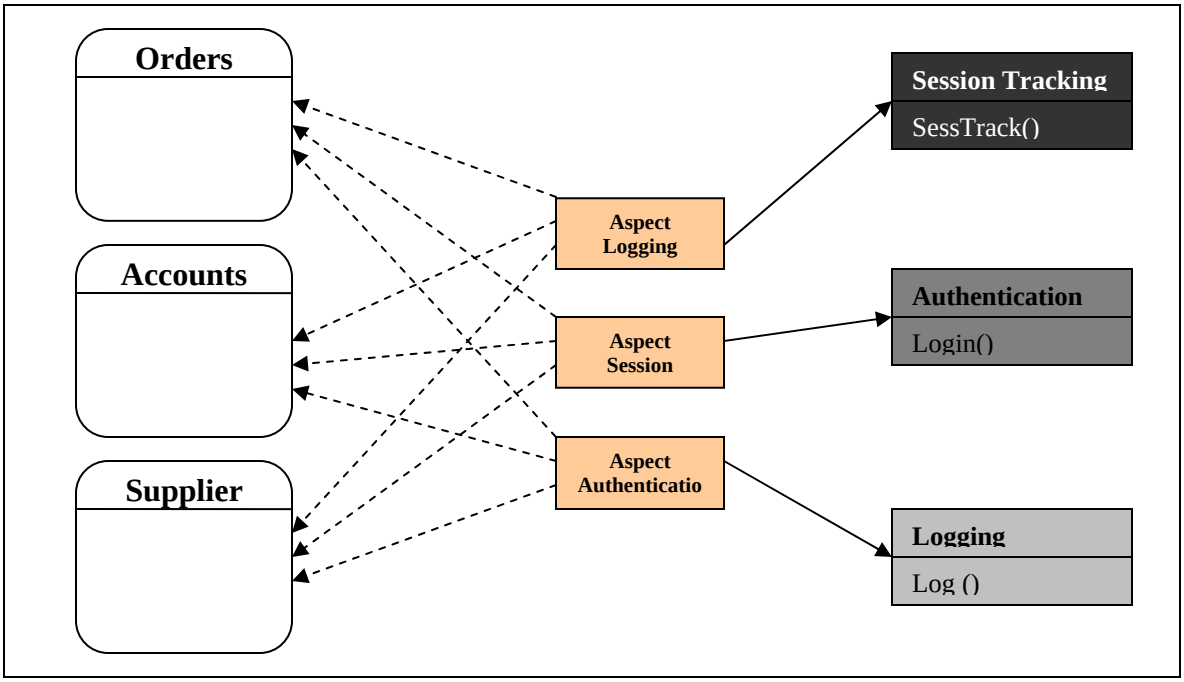


Figure 2.0 After AOP, the cross-cutting code remains separate from original code

One of the most successful software development models is Object Oriented Model; it is an enormous accomplishment in the world of software development. The reason of its success is that it maps the problem near to the real world in form of objects; but still there is code redundancy in object oriented model.

To make object oriented model more efficient a new model named as Aspect oriented model was introduced. Aspect-oriented programming (AOP) is one of the solutions to the problem of creating clean, well-encapsulated objects without inappropriate functionality.

In some cases, object-oriented programming introduces or causes inefficiencies, and aspect-oriented programming helps in filling these gaps. The aim of Aspect-Oriented Programming (AOP) is not to replace Object-Oriented Programming (OOP), but to complement it, allowing you to create clearer and better structured programs.

2.0 Scope and Objective

In this tutorial, we shall cover the topic AOP with very brief description of terminology and to the point example. This tutorial does not deal with the complex examples of code. The main objective of this tutorial is to provide the fundamental understanding of AOP to the readers assuming that they have the basic concepts of OOP. We hope that by reading this tutorial you will be able to code complex programs with respect to Aspects. However if you exactly understand the brief description, you can easily code the complex one.

3.0 Motivation

Computer Science is evolving and probably this is the most dynamic field in Applied Sciences. Progress in the field of development is so fast that can only imagine. In the recent past AOP has emerged as a new way of programming. AOP is a very well-organized and efficient style of coding that resolves the complexity of OOP and made the code neat and clean. The other reason of writing this tutorial is that AOP is gaining the popularity among the developers in practice**.** Although it does not replace the existing OOP concepts but only improves them.
OOP programming is working well from its starting to date. It has enormous benefits over the old ones and approximately has replaced the all. Now, users found some difficulties and inefficiencies in OOP. They want to try some method to rid over these problems. Mainly they face the problem of using the same code again and again in class such as Logging, Profiling, Tracing, Session tracking, Session expiration and so on. The classes become complex using this type of code put together with the main code. For example if a class simply wants to access the database, required authentication, logging, and session tracking code in it. Now these all problems are solved by AOP using a single aspect. With the help of AOP, we write a simple class that only has code to access the database. All other functionalities (authentication, logging and session tracking) handle by a single aspect. In this way it makes the code concise and easy to handle

4.0 History

Gregor Kiczales is one of the true founders of AOP, currently working at the University of British Columbia. He worked on AOP at the Xerox Palo Alto Research Center (PARC) (1984-1999), he with his team members developed AspectJ most popular general-purpose AOP package) and launch it in the market in 2001. He explicates that in Object Oriented languages any complicated system could be created. According to him, they have found many programming problems for which neither OO programming or procedural techniques are not sufficient, such problems forces the implementation of those design paradigms that scattered throughout the code and resulting in tangled code for problem solving. Development and maintenance of such paradigms are exclusively difficult. IBM's also launched HyperJ in 2001 which is more powerful but less usable [1].

5.0 What are Aspects

We want to write a program/application for handling book records in a library, the core concerns of this program are labeling and indexing of books, while authentication and logging would be required to update the database. The code of Logging and Authentication is injected in the original program by creating the objects. These types of code such as Logging and Authentication are called cross-cutting concerns. Cross-cutting concerns are also called the aspects of a program that affect other concerns [2, 3].
Daily life examples of cross-cutting concerns

- Ball Points (Open(),Writing(),Close())
- Markers (Open(),Writing(),Close())
- Doors (Open (), mainFunc(), Close())

All three example have two common functions such as Open() and Close(). These common functions are called cross cutting concerns/aspects.
Also another example is given to understand the cross cutting concern.

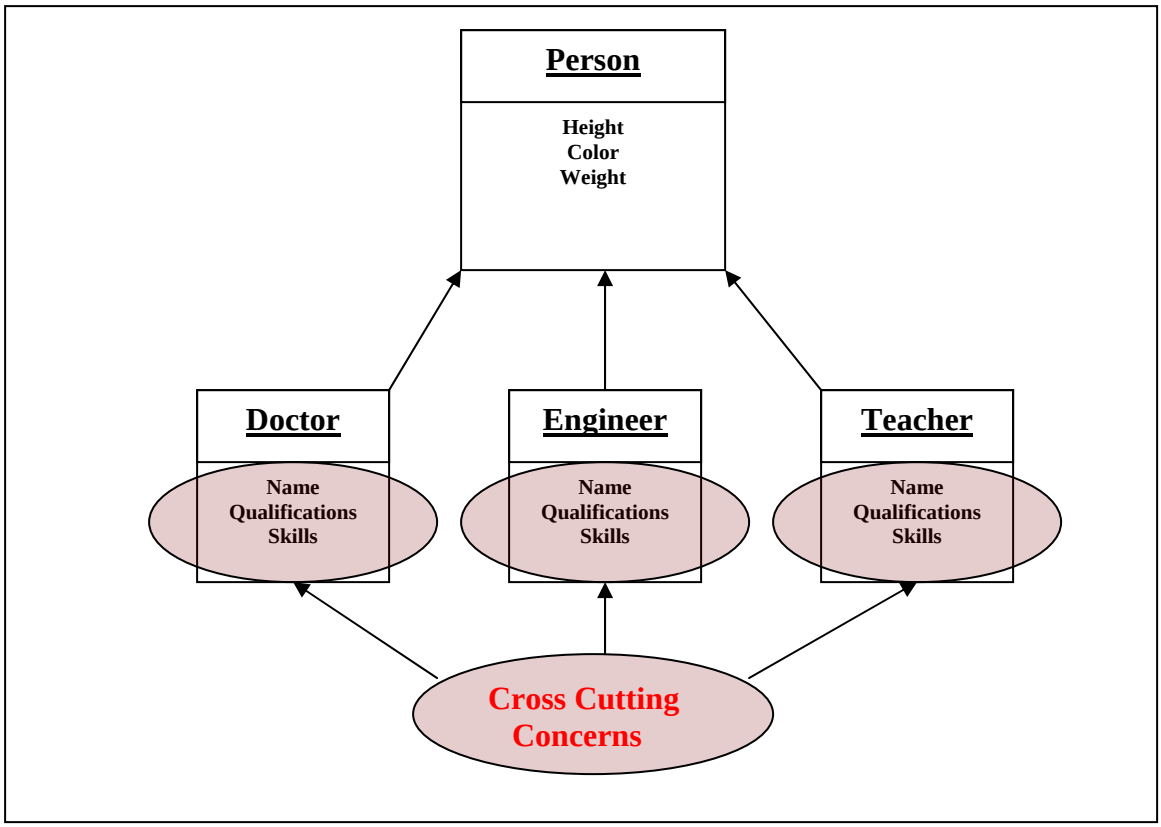


Figure 3.0 cross-cutting concerns

Some general types of cross cutting concerns/aspects are:

- Logging
- Profiling
- Error handling
- Authorization
- Performance monitoring
- Tracing
- Session tracking
- Special security management and so on

6.0 AOP Important Features

- AOP supports to develop applications with clean and manageable code
- AOP makes the maintenance process easier.
- AOP supports OOP Programming to solve problems
- AOP controls the concurrent code using Aspects and makes the code concise

7.0 Elements and Terminology

Following are the basic elements of AOP:

- Join points
- Pointcuts
- Advice
- Aspects
- Intertype Declaration

7.1 Join points

These are the points during an execution of a program where we can apply crosscutting code. Usually following points are called 'join points' where we want to call our cross cutting code.

Join points are:	Join points are not:
When a method/constructor call or execution	Execution of loops
initialization of a class or object	super calls
field read and write access	throws clauses in exception handling
exception handlers	multiple statements in a class

Table 1.0 Joint points

7.2 Pointcuts

We declare pointcut to specify where we want to apply our crosscutting code. A pointcut can apply on a set of join points. Following are the pointcuts:

Pointcuts	Description
call(*signature*)	Apply to join points of calling a specific method or constructor
execute(*signature*)	Apply to join points of executing a specific method or constructor
get(*signature*)	Apply to join points of reading the specific field
set(*signature*)	Apply to join points of writing the specific field
handler(*type-pattern*)	Apply to join points of exception handler related to the Throwable type-pattern

Table 2.0 Pointcuts

7.3 Advice

An advice combines the code that we want to apply, with the join points selected by our declared pointcut. We put the code inside the advice and then state when it executed with respect to the matched join point. There are three general types of advices.

Advice	Description
before()	Execute the code before the selected join point(s)
after()	Execute the code after the selected join point(s)
around()	Execute the code at the join points, letting us wrap or skip the execution of the join point(s) if required

Table 3.0 Advices

7.4 Inter-type declarations

AspectJ enables us to add new members, methods and fields to an existing Java class or type. These supplementary declarations, enclosed inside aspects, are called inter-type declarations. The newly declared members appear as they are directly implemented by the original class or type.

7.5 Aspects

Aspects are defined in the same way as we define a class.
An aspect has these things in it.

1. Pointcuts
2. Advice
3. Inter-type declarations

Like Java classes, aspects can have

- fields and methods (both static and non-static)
- abstract aspects
- extend aspects, to create new aspects

8.0 Example

Here is a simple example to understand the working of AOP.

This is a simple java program

HelloWorld.java

```
        public class HelloWorld
        {
   public static void say(String msg)
                {
                System.out.println(msg);
                }
   public static void sayTo(String msg, String name)
                {
                System.out.println(name + ", " + msg);
                }
        }
```

This is a test class to test the HelloWorl.java class

Test.java

```
        public class Test
        {
   public static void main(String[] args)
                {
                  HelloWorld.say(“Good Morning”);
                  HelloWorld.sayTo(“Good Morning”, “Asif”);
                }
        }
```

This is the output of program Test.java

Good Morning
Asif, Good Morning

This is an aspect

This is a pointcut name greetings. It defines that whenever a function name contain "say" at start is called of Hello World class, you can do something before and after this call that is defined in advice

GreetingsAspect.java

```
public aspect GreetingsAspect
        {
           pointcut greetings() :
                   call(public static void HelloWorld.say*(..));

           before() : greetings()
                 {
              System.out.print("Hello!");
            }

           after() : greetings()
                 {
              System.out.println("Thank you!");
            }
        }
```

This is an advice which instructs to print 'Hello!' before the 'greetings()' pointcut and 'Thank you' after the 'greetings()' pointcut.

This is the output of GreetingsAspect.java

Hello! Good Morning
Thank you!
Hello! Asif, Good Morning
Thank you!

9.0 Tools

Most widely used IDEs have option to AOP such as Eclipse, MyEclipse, IntelliJ etc. There are also many programming languages that implement AOP such as Java (AspectJ), .NET Framework (C#/ VB.NET), Haskell etc.
These tools are mostly used for AOP [3]

- **AspectJ**
- **AspectWerkz**
- **JBoss AOP**
- **Spring AOP**

10.0 References:

[1] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M.Loingtier, and J. Irwin, “Aspect-Oriented Programming”, ECOOP’97, June, 1997

[2] Mahoney J. V., *Functional Visual Routines*, Xerox Palo Alto Research Center, Palo Alto SPL95-069, July 30, 1995, 1995.

[3] Mendhekar A., Kiczales G., et al., *RG: A Case-Study for Aspect-Oriented Programming*, Xerox PARC, Palo Alto, CA. Technical report SPL97-009 P9710044, February, 1997.