Introduction to AOP

AOP is concept which has been supported by String. its a modern way to write our programs in efficient way.

Before we start with AOP lets have some discussion about the approaches we followed earlier and why we moved to AOP.

#### Functional Programming

Approach:

While developing our application we generally divide our business requirement in small pieces and write functions for each of them.

Function call other function and talk to each other, this is the way function complete the tasks.

Function Q

Function Z

Function A

Function B

Function H

Function D

Function C

Problem:

Problem with this solution is that if we have huge problem statement than we can have lots of function in our application and our application might become messy.

Data can travel easy anywhere which can be a security breach and our data is not secure here.

Solution to this problem:

We have solution to this problem ie we can go with Object Oriented approach to develop our application.

In Object Oriented approach we have classes which can define our entities with data member and member function with more cleaner approach which is easy to manage.

Object C

Procedures

Member Variable

Object B

Object A

Member Variable

Member Variable

Procedures

Procedures

Problem with Object Oriented programming:

Lets say we have to implement logging in our project and we have included it in each of our modules.

See the below diagram for reference:

Object C

Functions

.

logMessage()

.

Other Functions

Functions

.

logMessage()

.

Other Functions

Object B

Object A

Functions

.

logMessage()

.

Other Functions

We have logMessage in each class or module but is it a good approach to include logging or some other functionality to our modules which has nothing to do with business or business logic.

I think no…

We should not include the code in our business logic which has nothing to do with business, it can hamper performance and can increase maintaining cost of our application.

We can separate this logging from our application by creating another module/function/static function/creating and including it in super class.

Object C

Functions

.

logMessage()

.

Other Functions

Functions

.

logMessage()

.

Other Functions

Object B

Object A

Functions

.

logMessage()

.

Other Functions

Functions

.

logMessage()

.

Logger

Now where ever Object A,B and C will require the logging. Will create an object for Logger and can call logMessage() method.

But again we will have a problem of dependency of logging in our application, modules in our application will have lots of Logger Objects(depedencies) which has no relevance in our application.

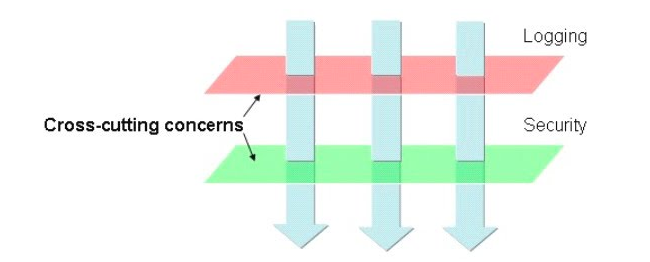
Problems:

* Too many relationships to the crosscutting objects
* Code is till enquired in all methods
* Can’t all be changed at once

#### Cross Cutting Concerns:

Concerns are two types:

1. The concerns representing single and specific functionality for primary requirements are known as **core concerns**.  
   OR  
   Primary functionlity of the system is knows as core concerns.  
   **For example**: Business logic
2. The concerns representing functionalities for secondary requirements are referred to as **crosscutting concerns or system-wide concerns**.  
   OR  
   The **crosscutting concern** is a concern which is applicable throughout the application and it affects the entire application.  
   **For example:** logging, security and data transfer are the concerns which are needed in almost every module of an application, hence they are cross-cutting concerns.



This figure represents a typical application that is broken down into modules. Each module’s main concern is to provide services for its particular domain. However, each of these modules also requires similar ancillary functionalities, such as security logging and transaction management. An example of crosscutting concerns is "logging," which is frequently used in distributed applications to aid debugging by tracing method calls. Suppose we do logging at both the beginning and the end of each function body. This will result in crosscutting all classes that have at least one function.

Example:

* Logging
* Transactions
* Security

#### Aspect Oriented Programming (AOP)

Object C

Functions

.

~~logMessage()~~

.

Other Functions

Functions

.

~~logMessage()~~

.

Other Functions

Object B

Object A

Functions

.

~~logMessage~~()

.

Other Functions

Aspect Configuration

Transactional Aspect

Logging Aspect

Here we have Object A,B and C with some methods say X,Y and Z in Object A and P,Q in Object B

We also have Logging Aspect defined separately.

Now we have to implement logging in A,B and C methods of Object A and P,Q methods of Object B

Instead of writing code/creating objects in our module we have Aspect Configuration where we can configure our methods A,B,C and P,Q to use logging aspect

Spring Container will read Aspect Configuration and inject Logging Aspect to A,B,C and P,Q.

Benefit of this approach is we have logging separate from our actual business functionality which is independent from our code.

If any point of time we have to make some changes in logging functionality, without touching our business logic or code we can make changes done in logging aspect only.

Object C

Object B

Object A

Functions

.

~~logMessage~~()

.

Other Functions

Functions

.

P()

Q()

.

Other Functions

Functions

.

A()

B()

C()

.

Other Functions

**inject**

Aspect Configuration

Transactional Aspect

Logging Aspect

Functions

A()

{

}

.

We can also wrap our aspects to the target methods also. lets say we have one target method and we want to execute our aspect to get executed before and after target method. We can achieve the same using around aspect functionality of AOP.

I will discuss this functionality in my subsequent posts.

Aspects are like Triggers in mysql

and Interceptors in Servlets

Aspect Code

Target Code

Aspect Code

#### Steps in AOP

* Write Aspects
* Configure where the aspects apply.