SPRING AOP

LEAVE A COMMENT

AOP stands for Aspect-Oriented Programming and it complements Object-Oriented Programming (OOP), providing another way of thinking about program structure. The fundamental unit of modularity in OOP is the class, while in the POA of the unit of modularity is the look. Aspects enable the modularization of concerns such as transaction management that span multiple types and objects.

One of the key components of spring is the AOP framework. While the Spring IOC container does not depend on AOP, meaning that there is no need to use AOP if you do not want to; AOP complements Spring IOC to provide a very capable middleware solution.

Uses of AOP spring frame works are,

1. It provides declarative enterprise services, particularly as a replacement for EJB declarative services. The most important such service is declarative transaction management.
2. It helps to allow users to implement custom aspects, complementing their use of OOP with AOP.

AOP terminologies:-

Spring uses different types of terminologies. They are,

1. Aspect
2. Joint Point
3. Advice
4. Pointcut
5. Introduction
6. Target Object
7. AOP Proxy
8. Weaving

Aspect

One aspect is a subprogram that is associated with a specific feature of a program. It is a feature or characteristic that crosses over the object.

AOP also enhances the modularity of a program.One aspect is scattered under the function is read over a number of related functions. That means that the change in a function requires modifying all affected modules. As the code making it more difficult to understand and maintain.

Modularization is a concern that crosses several classes. Transaction management is a good example of a crosscutting concern in J2EE applications. In Spring AOP, aspects are implemented using regular classes or regular classes annotated with @ aspect annotation (@ AspectJ style).

Eg:-

```
public void businessOperation(BusinessData data){
// Logging
logger.info("Business Method Called");
// Transaction Management Begin
transaction.begin();
// Do the original business operation here
transaction.end();
}
```

Joint point

A junction point is a point used in the spring of AOP to represent a framework for implementing the method. It always point during program execution or methods or exceptions. It is actually from the methods, AOP interception at this time to do some other things. ie early DoThis methods (), doExtra (), test (), somethingElse () etc are point during the execution of a program, such as the execution of a method or exception handling. In Spring AOP, a junction point is always a method of execution.

Eg:-

```
public void someBusinessOperation(BusinessData data){
//Method Start -> Possible aspect code here like logging.
try{
// Original Business Logic here.
}catch(Exception exception){
// Exception -> Aspect code here when some exception is raised.
}finally{
// Finally -> Even possible to have aspect code at this point too.
}
// Method End -> Aspect code here in the end of a method.
}
```

Advice

A tip or advice is an action taken by the look in a particular place of articulation. This is a piece of code that runs during program execution. There are several types of councils, depending on the position to be called a program, as before, after, around the action, so taken by one particular aspect, join point. Different types of advice include “around,” “before” and “after” advice. Many AOP frameworks, including spring, model an advice as an interceptor, maintaining a chain of interceptors around the junction.

Types of Advices

- Before Advice: – The advice which executed before a joinpoint called before advice. The before advice does not have the ability to interrupt the execution flow proceeding at the joint point unless it throws an exception.
- After return Advice: – The advice which executed after a jointpoint completed normally, called after advice. For example a method returns without throwing an exception.
- Around Advice: – Around advice is responsible for choosing whether to proceed to the joinpoint or to shortcut executing by returning their own return value or throwing an exception. It performs the custom behavior before and after method invocation by surrounding a joinpoint.
- After Throwing Advice: – This advice is executed when a method throws an exception.
- After (finally) Advice: – The advice is executed when program exits the joinpoints normally or by throwing an exception.

Pointcut

A predicate that matches binding sites and boards are associated with a cutoff expression and runs on any point of attachment corresponds to the cutoff (eg, execution of a method with a certain name). The concept of join points as the cutoff is essential together with expressions for Spring AOP and AspectJ language uses the expression cutoff by default. In AOP a cut is a set of joinpoints many that a council can run. When program execution reaches the point of a structure described in the cut-off, a piece of code (known as travelers) associated with that breakpoint executed. This provides additional flexibility for the programmer to combine additional code that runs on the point is already defined. This is a key concept of AOP, which distinguishes it from larger supply of interception technology.
Spring supports union and intersection operation setpoint. Union, the method or the cut-off matches. Pointcut is more useful. And Intersection means the methods that both cutoffs coincide. For advice on all kinds of advice is general advice used in AOP.

Introduction

Spring AOP allows to introduce new interfaces (and a corresponding application) to any object advises. For example, you could use an introduction to a grain of IsModified implement an interface, to simplify caching. An introduction is known as an inter-type declaration in the AspectJ community.

Target Object

An object is assisted by one or more respects. Also known as the object advised. Since Spring AOP is implemented using runtime proxies, this object will always be a proxy object. Objects in the spring framework, which is advised by one or more aspects are called the target object. These objects will always be subject to proxy because Spring AOP is implemented using runtime proxies.

AOP Proxy

AOP proxy is an object used to perform the contract area. This object is created by the AOP framework. In Spring AOP proxy is part of JDK dynamic proxy or proxy CGLIB.

Weaving

Aspects linked with other applications or objects to create an object advice. This can be done at compile time (using the AspectJ compiler, for example), load time or at runtime. Spring AOP, like other pure Java AOP frameworks, performs runtime tissues. The fabric is a process of linking issues with the application or object to create a type of counseling. This can be accomplished / performed at compile time, run time load time. In the spring framework tissue takes place at runtime.

Related Tutorials :

1. Spring Interview Questions
2. Spring Framework
3. Java Aspects
4. Spring Exception Handling
5. Struts to Spring

G+

0
Like