Java stories.

Entries RSS | Comments RSS

Search

- March 2009 (12)
- February 2009 (8)
- January 2009 (7)
- December 2008 (3)
- November 2008 (6)
- September 2008 (6)
- June 2008 (1)
- May 2008 (2)

Recent Comments

- mani on Spring AOP :Pointcut in details
- Tweets that mention How to install python plugin for eclipse :Pydev tutorial « Java stories. -- Topsy.com on How to install python plugin for eclipse :Pydev tutorial
- Harshit on Swing : Use JTable to display a List of Objects
- Cillia johnson on Swing : Use JTable to display a List of Objects
- Susan Yates on How to install python plugin for eclipse :Pydev tutorial

Spring AOP tutorial -I

Posted on May 18, 2009 by Harshit
7 Votes

Aspect-Oriented Programming (AOP) complements Object-Oriented Programming (OOP) by providing another way of thinking about program structure. The key unit of modularity in OOP is the class, whereas in AOP the unit of modularity is the aspect. Aspects enable the modularization of concerns such as transaction management that cut across multiple types and objects.

AOP concepts:

• **Aspect**: a modularization of a concern that cuts across multiple classes. Transaction management is a good example of a crosscutting concern in J2EE applications. In my words: a trigger which can affect the multiple classes a one point….
• **Join point**: a point during the execution of a program, such as the execution of a method or the handling of an exception. In Spring AOP, a join point always represents a method execution. In my words: a locus of points where execution will happen.
• **Advice**: action taken by an aspect at a particular join point. Different types of advice include “around,” “before” and “after” advice. (Advice types are discussed below.) In my words : the action to be taken at the point.
• **Pointcut**: a predicate that matches join points. Advice is associated with a pointcut expression and runs at any join point matched by the pointcut (for example, the execution of a method with a certain name). The concept of join points as matched by pointcut expressions is central to AOP, and Spring uses the AspectJ pointcut expression language by default. In my words a criteria used to locate point.
• **Introduction**: declaring additional methods or fields on behalf of a type. Spring AOP allows you to introduce new interfaces (and a corresponding implementation) to any advised object. For example, you could use an introduction to make a bean implement an IsModified interface, to simplify caching. (An introduction is known as an inter-type declaration in the AspectJ community.)
• **Target object**: object being advised by one or more aspects. Also referred to as the advised object. Since Spring AOP is implemented using runtime proxies, this object will always be a proxied object.
• **AOP proxy**: an object created by the AOP framework in order to implement the aspect contracts (advise method executions and so on). In the Spring Framework, an AOP proxy will be a JDK dynamic proxy or a CGLIB proxy.

Consider the example:

Lets declare an interface:

```
public interface Foo {

Foo getFoo(String fooName,int age);

void getAfter();

void getBefore(String myName);

}
```

A class implementing the interface:

```
public class DefaultFooService implements FooService {

      public Foo getFoo(String name, int age) {
            return new Foo(name, age);
      }
      public void getAfter() {}
      public void getBefore(String myName) {}
}
```

Till here we have simple java implementation.
Now let see come AOP concepts in picture.
Before - Now I want that before the getBefore() method is called I want to log message saying what is the parameter passed.
After - Also I want that once any method in the interface is called a message should be logged after it.
I have a class which will be called to satisfy the above criteria.

```
public class SimpleProfiler {

   public void afterMethod() throws Throwable {
         System.out.println("After the method call");
   }
   public void beforeMethod(String myName){
         System.out.println("My name is "+myName);
   }

}
```

The afterMethod() will log message after any method is called and beforeMethod() will log message before getBefore() is called.
To configure this we will used xmlThis is how I configure my spring.xml.

```
 <beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:aop="http://www.springframework.org/schema/aop"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">

   <!- this is the object that will be proxied by Spring's AOP infrastructure ->
1   <bean id="fooService" class="DefaultFooService"/>
2
3  <!- this is the actual advice itself ->
4   <bean id="profiler" class="SimpleProfiler"/>
5
6   <aop:config>
7               <aop:aspect ref="profiler">
14          <aop:pointcut id="aopafterMethod"
                  expression="execution(* FooService.*(..))"/>
15          <aop:after pointcut-ref="aopafterMethod"
                 method="afterMethod"/>

16         <aop:pointcut id="aopBefore"
                  expression="execution(* FooService.getBefore(String)) and
args(myName)"/>
17          <aop:before pointcut-ref="aopBefore"
                 method="beforeMethod"/>
         </aop:aspect>
   </aop:config>

</beans>
```

Let see how we have configure the AOP .

- Line 1 is used to create a proxy AOP object..
- Line 7 we define the aspect "SimpleProfiler" class which will come into picture at different point-cuts.
- Line 6 is used to configure the AOP.
- Line 14 defines a pointcut in which an expression needs to mention. In this case the expressions say that "call afterMethod of SimpleProfiler class for any method declared inside the FooService interface.
- Note Line 14 doesn't define when to call afterMethod().This is done in line 15
- Line 15 states that call afterMethod() for id aopAfterMethod

Similarly for beforeMethod we define in Line 16,17.

In above example we have

Aspect – SimpleProfiler.class
Point-cut – aopafterMethod,aopBefore
Advice <aop:before> <aop:after>

Now I am ready to run my main class and class methods of FooService.

```
 public class Boo {
public static void main(final String[] args) throws Exception {
BeanFactory ctx = new ClassPathXmlApplicationContext("spring.xml");
FooService foo = (FooService) ctx.getBean("fooService");
foo.getFoo("Pengo", 12);
foo.getAfter();
foo.getBefore("Harshit");
}
}


OutPut is
After the method call ( log messagefor getAfter method )
My name is Harshit (log message for getBefore)
After the method call (log message for getAfter method)
```

How to build various pointcut expression.

Possibly related posts: (automatically generated)

- **Spring AOP :Pointcut in details**
- **What is difference between == and equals in java?**
- **How to inject prototype dependency in a singleton bean**

Filed under: spring | Tagged: aop, spring

« Java Serialization algorithm revealed How to solve "error could not find java runtime 2 environment" while opening an IDE »

12 Responses

1. 
   Foo-1, on May 20, 2009 at 6:37 pm Said:

   Do you mean the interface FooService in the code:
   public interface Foo { ….

   Foo is a concrete class, isn't it?

   Reply
2. 
   Foo-1, on May 20, 2009 at 6:59 pm Said:

   There are a few typos: which parameter is called and we will used xml, etc…

   Question: in your xml line 13:

   What and where is the method "profile"?

   Reply
   - 
     rastogha, on May 21, 2009 at 7:07 pm Said:

     thanks for pointing the error.
     You need to create a dumy class Foo.
     I have removed some code not relevant in xml

     Reply
3. 
   Mahesh, on June 10, 2009 at 1:51 pm Said:

Can you please put workable example, it will be grately help in understanding this example.

Reply

4. 

Badal Chowdhary, on June 16, 2009 at 3:31 pm Said:

Nice One dude

Reply

5. **Spring AOP :Pointcut in details « Java stories.**, on June 23, 2009 at 11:19 am Said:

[...] Spring AOP tutorial -I [...]

Reply

6. 

Sam, on August 28, 2009 at 5:27 pm Said:

args(myObj) forces me to name variable as myObj

If the exact name of the argument (which is myObj) from aroundUpdate is passed in as args(myObj) in my pointcut, then aroundUpdate successfully gets the corresponding object inside the body of aroundUpdate method.
However, this forces the classes that that get intercepted by aroundUppdate method to name variable as myObj in the update method.

For example, MyDao.update method has to have method signature as MyDao.update (IDomain myObj). In other words. myObj has to be the name of the argument for it to work. I cannot arbitrarily name it as apple, oragange, instead of myObj.

The configuration shows below works, as long as the update method of classes have myObj as the name of the argument

Code:

```
public Integer aroundUpdate(ProceedingJoinPoint pjp, IDomain myObj) throws Throwable{

// code
}

MyDao{
public Integer update(IDomain myObj){
// code
}
}
```

Is there any way that I can use the fully qualified name of the IDomainObject (instead of myObj) in the above point cut expression?

If I could use IDomainObject in args, as opposed to myObj in the pointcut and could get a handle of the object inside the body of atroundUpdate, that would have solved my purpose.

When I try putting IDomainObject in pointcut as args(org.IDomainObject) and keep update method in MyDao as update(IDomain myObj), I get excception as:

Code:

```
org.springframework.beans.factory.BeanCreationException: Error creating bean with name
'_protectPointcutPostProcessor': BeanPostProcessor before instantiation of bean failed; nested
exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with
name 'org.springframework.aop.aspectj.AspectJPointcutAdvisor#1': Instantiation of bean failed;
nested exception is org.springframework.beans.BeanInstantiationException: Could not instantiate bean
class [org.springframework.aop.aspectj.AspectJPointcutAdvisor]: Constructor threw exception;
nested exception is java.lang.IllegalArgumentException: error at ::0 formal unbound in pointcut
at
org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory$1.run(AbstractAutowireCapableBeanFactory.java:405)
at java.security.AccessController.doPrivileged(AccessController.java:219)
```

According to Spring documentation, I can do so.

There is a similar proof in:

HTML Code:
http://denis-zhdanov.blogspot.com/2009/07/spring-aop-filtering-invocations-by.htmlIdeal situation would be to create pointcut as args(org.IDomain) because evey update method of *Dao does take an instance of IDomain as an argument. But I get exception when I try doing so, as shown above.

My goal is NOT to be forced to put the name of the argument as myObj for every update method of Dao that needs to invoke this point cut.

Could anyone please suggest what I am missing?

Could anybody suggest any variation in pointcut that will get me around?

Is there any other alternative?

Reply

7. 
Anup Jani, on October 13, 2009 at 5:54 am Said:

Very nice effort. Could be even simpler though. Might I add following Maven dependencies required:

junit
junit
3.8.1
test

org.springframework
spring
2.5.6

org.aspectj
aspectjweaver
1.6.4

org.aspectj
aspectjrt
1.6.4

Thank you.
Anup Jani
(Bsc, MCP, GNIIT, SCJP, SCWCD, IBM OOAD/UML JCert.)

Reply

8. 
anand, on October 29, 2009 at 2:36 am Said:

very helpfull…. thanks!!! made my day.

Reply

9. 
sivakumaran kathamutthu, on November 14, 2009 at 9:40 am Said:

coool tutorial . this very good tutorials for the new poeple of who like to learn AOP

Reply

10. 
radhakrishna, on November 27, 2009 at 11:14 am Said:

Nice one dude! thanks a lot

Reply

11. **links for 2010-02-17 | Tech Twin**, on February 17, 2010 at 6:02 pm Said:

[...] Spring AOP tutorial -I « Java stories. Nice AOP tutorial for starters (tags: AOP, Tutorial, Starters) [...]

Reply

Leave a Reply

Name

E-mail

Website

Submit Comment

Notify me of follow-up comments via email.

Notify me of site updates

- ## Top Posts
  - Spring AOP tutorial -I
  - how to access properties file in Spring
  - Use of hashcode() and equals()
  - Spring AOP :Pointcut in details
  - What is difference between == and equals() in java?
- Google™

  6 readers BY FEEDBURNER

  sitemeter
- ## Blogroll
  - Blog catalog
  - Code Monkeyism
  - http://tech.puredanger.com/
  - http://technoticles.com/
  - Java Interview questions
  - Technoticles
- ## Java stories.
  - Swing : Use JTable to display a List of Objects
  - How to install python plugin for eclipse :Pydev tutorial
  - IntegerCache in JDK1.5
- ## Pages
  - About Me
- ## Blog Stats
  - 59,277 hits