AOP里面3个概念Advice，PointCut，Advisor

处理逻辑（Advice）：在某个连接点所采用的处理逻辑。

切点（PointCut）：一系列连接点的集合，它指明处理方式（Advice）将在何处被触发，可以使用正则表达式表达。

Advisor：是PointCut和Advice的综合体，完整描述了一个advice将会在pointcut所定义的位置被触发。

附注：
AOP全名Aspect-Oriented Programming，我们先来看看XXX-Oriented的意义，通常翻译「XXX导向」，也就是以XXX为中心，例如中文中「客户导向」就是以客户为中心，而「对象导向」（OOP：Object-Oriented Programming）就是以对象为中心的程序设计。
自然的，Aspect-Oriented Programming，就是「Aspect导向程序设计」，也就是以Aspect为中心的程序设计，但什么是Aspect？中文直译通常是「方面」，但这个名词容易使人混淆。
牛津字典中的英英解释对Aspect是：particular part or feature of sth being considerd.
所以Aspect在英文中不只有「方面」的意思，还有部份（part）的意思。中文中称「就这个方面来说」，通常指的是「就这个角度来说」或「就这个方向来说」，这个解释不适用于AOP中的Aspect。如果英文中说from this aspect of sth，除了可以翻译为上面两句的意义之外，还可以翻作「就这个部份来说」。
以我们的前一个主题中的记录（log）动作插入至HelloSpeaker对象的hello()中为例，我们说「就记录这个部份」是不属于HelloSpeaker职责的，它被硬生生切入HelloSpeaker中，英文中我们可以说：The logging aspect of the "hello" method doesn’t belong to the job of HelloSpeaker.
所以以整个方法的执行流程来说，如果执行流程是纵向的，则记录这个动作硬生生的「横切」入其中，这个横切入的部份我们就称之为Aspect，它是横切关注点（crosscutting concern，一个concern可以像是权限检查、事务等等）的模块化，将那些散落在对象中各处的程序代码聚集起来。
所以Aspect要用中文表达的话，适切一些的名词该是「横切面」或「切面」。AOP关注于Aspect，将这些Aspect视作中心进行设计，使其中从职责被混淆的对象中分离出来，除了使原对象的职责更清楚之外，被分离出来的Aspect也可以设计的通用化，可运用于不同的场合。
来看事务管理这个Aspect如何在动态代理下被抽取出来，下面是一个简单的概念，基本上是在Handler中先激活事务，执行存储层动作，方法执行成功则提交（commit），失败则回滚（rollback）：
代码：

```
TransactionHandler.java
```

```
 public class TransactionHandler implements InvocationHandler {

private  Object delegate;


public  Object bind(Object obj) {

   this.delegate = obj;

   return Proxy.new ProxyInstance(.., .., ..);

}

public  Object invoke(Object proxy, Method method, Object[] args) {
```

```
        Object result = null;

        …..

        YourTransaction transaction = null;
        try {
            transaction = yourMethodForGettingTransaction();

            result = method.invoke(delegate, args);

            transaction.commit();
        }
        catch(YourException e) {
            if(transaction != null) {
                try {
                    transaction.rollback();
                }
                catch(Exception e) {}
            }
        }

        ….

        return result;
    }
}
```

在AOP中，有好几个关键的概念，然而其中更主要的是：PointCut、Advice与Advisor。这些术语并不是很直观，我们配合我们上一个主题的HelloSpeaker来说明会比较容易理解。

PointCut是JointPoint的集合，JointPoint是指Aspect加入的阶段点，例如某个方法被调用，某个成员被存取（Spring不支持），或是某个例外被丢出，以我们前一个主题的HelloSpeaker为例，hello()方法就是一个JointPoint。PointCut为JointPoint的集合，意味着多个方法或例外丢出可以使用同一个处理建议（Advice）。

Advice是在JointPoint上所要调用的处理建议（在JointPoint上所采取的动作，许多AOP框架通常以interceptor来实作Advice，之后会介绍），例如记录、事务处理、权限检查等。

Advice类型（advice type）有Around、Before、Throws、After returning。这些Advice

类型会分别在以下的时机被调用：在JointPoint前后、JointPoint前、JointPoint丢出例外时、JointPoint执行完毕后。

就我们上一个例子来说，我们在hello()这个JointPoint上会调用Around Advice，即我们的LogHandler。事务管理调用的Advice类型也是属于Around Advice，即上面的TransactionHandler，调用的JointPoint可能是某个DAO对象的saveXXX()方法。

Advisor将PointCut与Advice组合在一起，我们前一个主题中的HelloSpeaker例子很简单，故而没有使用到Advisor，之后会介绍到Advisor。

补充：动态代理是比较常见的AOP实作策略，在《Expert One-on-One J2EE Development WIthout EJB》 Rod Johnson、Juergen Hoeller中的第八章中有提到，AOP的实作有五个主要的策略：

- Dynamic Proxies
- Dynamic Byte Code Generation
- Java Code Generation
- Use of a Custon Class Loader

Language Extensions

归类于： 企业应用架构 — longer @ 7:58 pm 评论(0)

评论

该日志第一篇评论

发表评论

评论也有版权！

名称 (必填)

电子邮箱 (必填,不公开)

网址

提交评论

- **news**
  - 欢迎来这里讨论，共同进步，不断学习才能不断进步。