Alex Lin
2012-05-21

AJDT AspectJ CTW Maven Java

AspectJ 编译时织入(Compile Time Weaving, CTW)

本文主要介绍 AspectJ 编译时织入（Compile Time Weaving, CTW）的技术，并附有详细示例代码。

AspectJ 是一个 AOP 的具体实现框架。AOP（Aspect Oriented Programming）即面向切面编程，可以通过预编译方式和运行期动态代理实现在不修改源代码的情况下给程序动态统一添加功能的一种技术。

AspectJ不但可以通过预编译方式（CTW）和运行期动态代理的方式织入切面，还可以在载入（Load Time Weaving, LTW）时织入。

AspectJ 扩展了Java，定义了一些专门的AOP语法。官网上这样描述：

> a seamless aspect-oriented extension to the Java programming language Java platform compatible easy to learn and use

在网上，关于 Spring + AspectJ Annotation动态代理或者AspectJ 载入时织入（Load Time Weaving, LTW）的文章特别多（其特点是：前者需要用Java编写切面并加以注释，后者需要编写 aop.xml 文件并在启动 Java 时带上参数-javaagent），所以本文就不在涉及。
编译时织入是 AspectJ 的一个重要功能，因为 AspectJ 有一个专门的编译器用来生成遵守 Java 字节编码规范的 Class 文件。

要完成代码通过 AspectJ 编译时织入，通常需要两步：

1. 编写 aspect 文件；
2. 使用 ajc 编译器结合 aspect 文件对源代码进行编译。

我们可使用两个工具来方便我们开发 AspectJ 程序：

1. Eclipse 插件 AJDT（AspectJ Development Tools），方便我们在 eclipse 环境下编写切面（AspectJ）并在编译源码时自动织入切面；[参考用法]
2. AspectJ compiler Maven Plugin，Maven 的 AspectJ 编译插件，同样可以在编写源码时将切面织入到字节码。

下面一步步来创建一个示例。 首先，通过 mvn archetype:generate 来创建一个普通的maven工程。

```
mvn archetype:generate
```

在出来的列表中选择 maven-archetype-quickstart 即可。
然后，修改 pom.xml，增加 aspectj 相关内容，如下

```
            <outxml>true</outxml>
            <aspectLibraries>
            <!-- 此处定义外部的aspect包，例如spring的事务aspect包 。这里引用的包必须在依赖中声明
-->
            <!--
              <aspectLibrary>
                 <groupId>org.springframework</groupId>
                 <artifactId>spring-aspects</artifactId>
               </aspectLibrary>
            -->
          </aspectLibraries>
         </configuration>
         <executions>
           <execution>
            <goals>
             <goal>compile</goal><!-- use this goal to weave all your main classes --
>
              <!-- <goal>test-compile</goal> -->
            </goals>
```

近期文章

RingList，保留最后 N 项数据的环状数据结构
2015-08-29

VMware ESXi 6.0（88SE9230 Raid卡，RTL8111e网卡）安装注意事项
2015-05-25

通过 U 盘安装 VMware ESXi 5.5 的过程及其注意事项
2014-10-02

几种可以免费托管静态网站的服务对比
2014-07-31

将 OpooPress 静态博客发布到 GitCafe Pages
2014-07-31

使用 Apache SSL 反向代理 Google 搜索
2014-06-25

解决 Google Web Fonts 无法加载导致网页卡顿的问题
2014-06-06

近期评论

```
          </execution>
        </executions>
    </plugin>

    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-eclipse-plugin</artifactId>
      <version>2.6</version>
      <configuration>
         <ajdtVersion>1.6.11</ajdtVersion>
      </configuration>
    </plugin>
   </plugins>


  </build>
```

通过命令创建Eclipse工程

```
mvn eclipse:eclipse
```

将项目 import 到 Eclipse。

创建一个 Service 接口(SampleService.java)，如下：

```
public interface SampleService {
    int add(int x, int y);
    String getPassword(String username);
}
```

其实现类 (SampleServiceImpl.java) 如下：

```
public class SampleServiceImpl implements SampleService{
        public int add(int x, int y) {
                return x + y;
        }

        @AuthCheck
        public String getPassword(String username) {
                return "password";
        }
}
```

下面我们定义切面(Aspect) (SampleAspect.aj 注意后缀是aj)

```
        public pointcut serviceAddMethods(): execution(public * org.opoo.samples.aspectj.SampleService+.add*(..));

        Object around(): serviceAddMethods(){
                Object oldValue = proceed();
                System.out.println("原值是: " + oldValue);
                return Integer.MIN_VALUE;
        }

        /**
         * 切入点: SampleService继承树中所有标注了AuthCheck的方法。
         */
        public pointcut serviceAuthCheckAnnotatedMethods(): execution(* org.opoo.samples.aspectj.SampleService+.*(..)) && @annotation(AuthCheck);

        before(): serviceAuthCheckAnnotatedMethods(){
                if(1==1){//权限检查代码
                        throw new IllegalStateException("权限不足");
                }
        }

        /**
         * 切入点: SampleService继承树中所有 public 的方法。
         */
        public pointcut serviceAllPublicMethods(): execution(public * org.opoo.samples.aspectj.SampleService+.*(..));


        after(): serviceAllPublicMethods(){
                MethodSignature methodSignature = (MethodSignature) thisJoinPoint.getSignature();
                Method method = methodSignature.getMethod();
                System.out.println("[LOG] 方法被调用了: " + method);
        }
}
```

第一个切入点是 SampleService 继承树中所有 public 且以 add 开头的方法。SampleServiceImpl#add(int,int) 方法满足这个条件。织入的代码是环绕织入，改变了被切入方法的返回值，不管输入参数如何都返回一个新值。
第二个切入点是 SampleService 继承树中所有标注了 AuthCheck 的方法，采用前置织入，模拟权限检查功能，本例中始终无权限，被织入方法调用时应该发生异常。

第三个切入点是 SampleService 继承树中所有 public 的方法。采用后置织入，模拟日志功能，记录方法调用情况。

然后通过 mvn 编译

```
mvn test-compile
```

编译后查看 target\classes 目录中的字节码文件，可以发现，SampleAspect.aj 也被编译成了 java 字节码。在编译过程中的输出信息可以看出切点织入情况：

```
[INFO] Join point 'method-execution(int org.opoo.samples.aspectj.SampleServiceImpl.add(i
nt, int))' in Type 'org.opoo.samples.aspectj.SampleServiceImpl' (SampleServiceImpl.java:
23) advised by around advice from 'org.opoo.samples.aspectj.SampleAspect' (SampleAspect.
aj:30)
[INFO] Join point 'method-execution(int org.opoo.samples.aspectj.SampleServiceImpl.add(i
nt, int))' in Type 'org.opoo.samples.aspectj.SampleServiceImpl' (SampleServiceImpl.java:
23) advised by after advice from 'org.opoo.samples.aspectj.SampleAspect' (SampleAspect.a
j:41)
[INFO] Join point 'method-execution(java.lang.String org.opoo.samples.aspectj.SampleServ
iceImpl.getPassword(java.lang.String))' in Type 'org.opoo.samples.aspectj.SampleServiceI
mpl' (SampleServiceImpl.java:31) advised by after advice from 'org.opoo.samples.aspectj.
SampleAspect' (SampleAspect.aj:41)
[INFO] Join point 'method-execution(java.lang.String org.opoo.samples.aspectj.SampleServ
iceImpl.getPassword(java.lang.String))' in Type 'org.opoo.samples.aspectj.SampleServiceI
mpl' (SampleServiceImpl.java:31) advised by before advice from 'org.opoo.samples.aspect
j.SampleAspect' (SampleAspect.aj:54)
```

注意：虽然 Eclipse 安装了 AJDT 插件，但有时候可能编译的字节码中并没有织入切面，所以建议还是执行 maven 命令进行编译。
编译完成后就可以在 Eclipse 中执行单元测试检验成果了。

为了有个更直观的印象，我们可以反编译类 SampleServiceImpl 来看看字节码中的切面织入情况。这里我们使用 jad 作为反编译工具，命令如下（在项目根目录即 pom.xml 文件所在目录执行）：

```
d:\path-to-jad\jad.exe -sjava -o -r -ff -d target\src target\classes\**\*.class
```

查看 target\src 目录反编译出的 SampleServiceImpl.java 文件，部分代码如下：

```
public class SampleServiceImpl implements SampleService
{
    public int add(int x, int y)
    {
        int i = x;
        int j = y;
        int k;
        try
        {
            k = Conversions.intValue(add_aroundBody1${'$'}advice(this, i, j, SampleAspec
t.aspectOf(), null));
        }
        catch(Throwable throwable)
        {
            SampleAspect.aspectOf().ajc${'$'}after${'$'}org_opoo_samples_aspectj_SampleA
spect${'$'}3${'$'}8192077e(ajc${'$'}tjp_0);
            throw throwable;
        }
        SampleAspect.aspectOf().ajc${'$'}after${'$'}org_opoo_samples_aspectj_SampleAspec
t${'$'}3${'$'}8192077e(ajc${'$'}tjp_0);
        return k;
    }
    //...
```

可以看出，字节码中已经织入了我们所定义的切面。

编译时织入总结

优点：

1. 由于将切面直接编译进了字节码，所以运行时不再需要动态创建代理对象，节约了内存呢和 CPU 消耗；
2. 通过AspectJ，方法被织入了切面后，方法上的 Annotation 还是有效的，因为对象类型没有变。而动态代理可能会以代理类替代原类型，也就失去了 Annotation。

缺点：

1. 编写 aspect 文件有一定的难度；
2. 编译过程稍显复杂（借助工具可简化：Eclipse AJDT, Maven AspectJ 插件等）。

完整示例代码下载

Source

« Spring+ActiveMQ: 嵌入式和独立运行ActiveMQ的配置示例
WordPress 邮件配置：解决在 SourceForge 空间安装 WordPress 无法发送邮件的问题 »

相关文章

Spring+ActiveMQ: 嵌入式和独立运行ActiveMQ的配置示例	2012-05-24
Java Object Query System(JOQS) 整理	2012-04-18
Java Object Query System(JOQS) 简介	2012-04-13
建立统一的存储接口	2013-12-17
Oracle Coherence 缓存总是返回新的对象实例	2014-03-17

评论