你必须遵守约定！Maven Enforcer Plugin

July 30th, 2010

在一个稍大一点的组织或团队中，你无法保证所有成员都熟悉Maven，那么他们做一些比较愚蠢的事情就会变得很正常，例如给项目引入了外部的SNAPSHOT依赖而导致构建不稳定，使用了一个与大家不一致的Maven版本而经常抱怨构建出现诡异问题，或者不小心引入了一个项目禁止使用的依赖或者插件，等等。如果你的项目有十几个或者更多的模块，团队成员也是两位数，隔三差五地遇到并不得不解决这种琐碎的问题显然非常烦人。当然，你可以引入Maven培训来加强大家的技术能力，从而避免类似的错误不断发生，但本文要介绍的是一种更便捷的做法，就是使用一个大家可能未听说过的Maven插件：Maven Enforcer Plugin。

我们知道，Maven的核心设计原则之一是“约定优于配置”，例如不进行任何配置，Maven就知道从src/main/java读取源码，从src/test/java读取测试代码并自动运行。Maven的超级POM其实就是约定的实现，在这里你能看到各种约定的配置，包括目录结构，核心插件版本，等等。所有的Maven POM都隐式地继承自超级POM从而获得这些配置。

在此基础上，你在组织内部可能有进一步的约定希望大家遵守，例如固定的JDK版本，固定的Maven版本，固定某些依赖的版本，固定某些插件的版本等等，这个时候Maven Enforcer Plugin就能派上用场。举个最简单的例子，我希望大家使用的JDK版本都是1.5及以上，Maven版本都是2.2.1及以上，那么就可以在父POM中如下配置Maven Enforcer Plugin：

```
1  <build>
2   <plugins>
3     <plugin>
4       <groupId>org.apache.maven.plugins</groupId>
5       <artifactId>maven-enforcer-plugin</artifactId>
6       <version>1.0-beta-1</version>
7       <executions>
8         <execution>
9           <id>enforce-versions</id>
10          <goals>
11            <goal>enforce</goal>
12          </goals>
13          <configuration>
14            <rules>
15              <requireMavenVersion>
16                <version>2.2.1</version>
17              </requireMavenVersion>
18              <requireJavaVersion>
19                <version>1.5</version>
20              </requireJavaVersion>
21            </rules>
22          </configuration>
23        </execution>
24      </executions>
25    </plugin>
26  </plugins>
27 </build>
```

上述代码中使用了Maven Enforcer Plugin的enforce目标，该目标会基于rules配置进行检验。这里我们配置两条rule，第一条是requireMavenVersion，值2.2.1表示大于等于2.2.1，第二条规则是requireJavaVersion，值1.5表示大于等于1.5。如果你想配置更为复杂的版本范围，可以参考该文档，例如 (1.0,2.0] 就表示大于1.0小于等于2.0。

除了requireMavenVersion和requireJavaVersion之外，Maven Enforcer Plugin内置了很多其它的rule，包括如何禁止某些依赖、设定操作系统版本、设定插件版本等等，读者可以在这里看到完整的内置rule的列表。下面就再举一个关于依赖的例子，假设你的项目全面使用了TestNG，因此你不想在依赖中看到JUnit，这个时候就可以使用bannedDependencies这条rule，如：

```
1      <plugin>
```

```
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-enforcer-plugin</artifactId>
        <version>1.0-beta-1</version>
        <executions>
          <execution>
            <id>enforce-banned-dependencies</id>
            <goals>
              <goal>enforce</goal>
            </goals>
            <configuration>
              <rules>
                <bannedDependencies>
                  <excludes>
                    <exclude>junit:junit</exclude>
                  </excludes>
                  <message>you must use TestNG</message>
                </bannedDependencies>
              </rules>
            </configuration>
          </execution>
        </executions>
      </plugin>
```

上述代码中bannedDependencies下有一条exclude配置，值为junit:junit，表示排除所有groupId为junit，artifactId为junit的依赖，而message用来配置当Maven检查到有人构建中有junit依赖事将打印的错误输出。

还有一条非常有用的rule是requireReleaseDeps，用来禁止项目引入SNAPSHOT依赖，具体配置不再赘述，读者可以直接参考官方文档。

如果所有这些内置的rule都无法满足你的需求怎么办，Maven Enforcer Plugin提供了开放的接口允许你编写自己的rule，有兴趣的读者可以参考这个文档。

OK，现在你只要在父POM里配置好Maven Enforcer Plugin，然后要求所有子模块来继承，你就能很好的控制项目的约定，如果再有人破坏约定，那么他们不得不面对Maven的错误提示，从而自觉修复这些问题。

最后，Youtube上还有两段Sonatype提供的关于Maven Enforcer Plugin的视频，有兴趣的朋友可以去看看：

- http://www.youtube.com/watch?v=bgLag8rxSJE
- http://www.youtube.com/watch?v=XVZKLZPewSg

原创文章，转载请注明出处，本文地址：http://www.juvenxu.com/2010/07/30/you-have-to-follow-convention-maven-enforcer-plugin/

Wang zi

August 7th, 2010 at 09:02 | #1 Reply | Quote

我问一个问题啊，为什么要使用maven

juvenxu

August 24th, 2010 at 20:34 | #2 Reply | Quote

关于这一点，可以看一篇我以前写的博客：http://juvenshun.javaeye.com/blog/250855

董越

November 1st, 2013 at 17:00 | #3 Reply | Quote

您好，想检查各个Maven聚合项目总是与其包含的子模块们共享同一个版本号，初步想法是仿照http://maven.apache.org/enforcer/enforcer-api/writing-a-custom-rule.html 编写一个custom-rule，但不知如何编程实现获得各子模块的版本号信息。请求指点，谢谢！

November 4th, 2013 at 15:09 | #4 Reply | Quote

juvenxu

可以参考这个：https://fisheye.codehaus.org/browse/mojo/tags/versions-maven-plugin-2.1/src/main/java/org/codehaus/mojo/versions/UpdateChildModulesMojo.java?r=18420