七、Tomcat热部署与热加载

热部署和热加载是类似的，都是在不重启Tomcat的情况下，使得应用的最新代码生效。
热部署表示重新部署应用，它的执行主体是Host，表示主机。
热加载表示重新加载class，它的执行主体是Context，表示应用。

Tomcat中的后台线程

热部署和热加载都需要监听相应的文件或文件夹是否发生了变化。它们都是由Tomcat的后台线程触发的。

BackgroundProcessor就表示后台线程。

每个容器都可以拥有一个BackgroundProcessor，但是默认情况下只有Engine容器会在启动的时候启动一个BackgroundProcessor线程。

该线程会每隔一段时间（可以设置，单位为秒），去执行后台任务，先执行本容器定义的后台任务，然后再执行子容器的定义的后台任务，子容器的任务执行完成后会继续执行其子容器的任务，直到没有子容器为止。**从这里可以看出就算每个容器自己开启一个BackgroundProcessor，也只不过是多了一个执行相同任务的线程而已，执行任务的效率有所提升。**

对于后台任务，所有容器会有一些统一的任务需要执行：

1. 集群服务器心跳
2. 如果一个容器拥有自己的类加载器，那么查看是否需要进行热加载
3. 检查Session是否过期
4. 执行每个容器对于的Realm对应的后台任务
5. 执行每个容器中pipeline中的每个valve的后台任务
6. 发布PERIODIC_EVENT事件

在这个过程中的**第2步中会触发热加载，第6步中会触发热部署**

热加载

我们可以在Context上配置reloadable属性为true，这样就表示该应用开启了热加载功能，默认是false。

热加载触发的条件是：**WEB-INF/classes目录下的文件发生了变化，WEB-INF/lib目录下的jar包添加、删除、修改都会触发热加载。**

热加载大致流程为：

1. 设置当前**Context不能接受以及处理请求标志**为true
2. 停止当前Context
3. 启动当前Context
4. 设置当前**Context不能接受以及处理请求标志**为false

我们着重来分析一下第2、3步。

我们不妨先来分析第3步-启动当前Context的过程中会发生什么事情：

1. 创建一个每个应用都单独自定义的**WebappClassLoader**
2. 解析web.xml文件，这一步会做很多事情，但是主要的目的是寻找定义的Servlet并把它添加到Context中去，而对于寻找Servlet需要进行两个方面的寻找，一是从web.xml中寻找定义的Servlet，二是从寻找class文件中添加了@WebServlet注解的类。大家很有可能认为，此时是不是会去加载我们定义的Servlet类，可以告诉大家的是，这个时候不会，Servlet类的加载是在后面步骤发生的，那么这里就有疑问了，我们要看一个类上是不是存在一个@WebServlet注解，应该要先加载这个类呀？Tomcat并没有这么做，它是直接先把class文件当做一个普通文件，然后看这个文件对应的地方是否存在一个WebServlet注解，如果存在，则认为这个class文件是一个Servlet，然后把这个class的全名封装到Servlet对象中去，然后将Servlet对象添加到Context对象中。在解析web.xml时也是类似了，对于我们定义的Servlet，最后都会生成一个Servlet对象，然后记录一个这个Servlet对象对应的class的全名，最后把Servlet对象添加到Context中去。
3. 我们在使用Servlet的时候还会用其他的一些注解比如@ServletSecurity、@RunAs等等，对于这些注解是有特定功能的，Tomcat为了识别这个注解，此时就要去真正加载我们的Servlet类了。当然要不要识别这些注解是可以配置的，如果不识别，那么这一步就不会发生了，那么Servlet类的加载就会在有请求过来时才会进行类的加载。

加载类过程：

1. 调用WebappClassLoaderBase的loadClass方法进行类的加载，该方法传递一个类的全限定名。
2. 要加载一个类，先得找到这个类在哪里，对应的是哪个classs文件，所以Tomcat中有一个缓存对象，该对象保存了一个类的全限定名对应的资源路径。当然，在第一次加载这个类时，这个缓存是空的，所以这个时候就要去寻找这个类对应的class文件地址，找到之后再缓存。接下来就来分析是怎么找到这个class文件地址的。
3. 其实查找很容易，现在WEB-INF/classes/目录下是否存在这个类，如果不存在就看WEB-INF/lib/目录下的JAR包中是否存在这个类，最终如果找到就将进行缓存，保存一个类的全限定名对应的class文件地址或jar包地址。

4. 当知道这个类在哪了之后，就可以defineClass了，最终得到一个class对象，并且也会将这个class对象设置到我们的缓存中，所以上文说的缓存中，其实是这么一个映射关系，**一个类的全限定名对应这个类的文件地址以及这个类的class对象。**
5. 所以当下次再有情况需要加载class时，就可以直接取缓存中的对应的class对象了。

这是第3步，我们在来看第2步：
对于第2步-停止当前Context，其实所做的事情比较单一，就是清空和销毁，而其中跟类加载相关就是清空上文中的缓存对象。

这样，我们的热加载就是先清空所有东西，然后重新启动我们应用，但是因为这个的触发条件基本上是class类发生了变化，所以热加载的过程中关于应用其他的一些属性是没有发生变化的，比如你现在想在Context中添加一个Vavle是不会触发热加载的，而如果要达到这个效果就要用到**热部署**。

> 注意：虽然我们在热加载的过程发现它是先停止再启动，做法看似粗暴，但是这样是性价比比较高的，并且这种方式至少比重启Tomcat效率要高很多。

> 注意：热加载不能用于war包

关于类的加载，这里有一点是需要注意的，对于一个class文件所表示的类，同一个类加载器的不同实例，都可以加载这个类，并且得到的class对象是不同的，回到热加载，我们举一个例子，我们现在有一个A类，一个自定义的WebappClassloader类，一开始先用一个WebappClassloader实例加载A类，那么在jvm中就会存在一个A类的class对象，然后进行热加载，先停止，再启动，在停止的时候会杀掉当前应用的所有线程（除开真正执行代码的线程），再启动时又会生成一个WebappClassloader实例来加载A类，如果热加载之前的那个A类的class对象还没有被回收的话，那么此时jvm中其实会存在两个A类的class对象，这是不冲突，因为class对象的唯一标志是**类加载器实例对象+类的全限定名**。

热部署

BackgroundProcessor线程第六步会发出一个PERIODIC_EVENT事件，而HostConfig监听了此事件，当接收到此事件后就会执行热部署的检查与操作。

对于一个文件夹部署的应用，通常会检查以下资源是否发生变动：
- /tomcat-7/webapps/应用名.war
- /tomcat-7/webapps/应用名
- /tomcat-7/webapps/应用名/META-INF/context.xml
- /tomcat-7/conf/Catalina/localhost/应用名.xml
- /tomcat-7/conf/context.xml

对于一个War部署的应用，会检查以下资源是否发生变动：

- /tomcat-7/webapps/应用名.war
- /tomcat-7/conf/Catalina/localhost/应用名.xml
- /tomcat-7/conf/context.xml

对于一个描述符部署的应用，会检查以下资源是否发生变动：

- /tomcat-7/conf/Catalina/localhost/应用名.xml
- 指定的DocBase目录
- /tomcat-7/conf/context.xml

一旦这些文件或目录发生了变化，就会触发热部署，当然热部署也是有开关的，在Host上，默认是开启的。这里需要注意的是，对于一个目录是否发生了变化，Tomcat只判断了这个目录的修改时间是否发生了变化，所以和热加载是不冲突的，因为热加载监听的是WEB-INF/classes和WEB-INF/lib目录，而热部署监听的是应用名那一层的目录。

在讲热部署的过程之前，我们要先讲一下应用部署的优先级，对于一个应用，我们可以在四个地方进行定义：

1. server.xml中的context节点
2. /tomcat-7/conf/Catalina/localhost/应用名.xml
3. /tomcat-7/webapps/应用名.war
4. /tomcat-7/webapps/应用名

优先级就是上面所列的顺序，意思是同一个应用名，如果你在这个四个地方都配置了，那么优先级低的将不起作用。因为Tomcat在部署一个应用的时候，会先查一下这个应用名是否已经被部署过了。

热部署的过程：
如果发生改变的是文件夹，比如/tomcat-7/webapps/应用名，那么不会做什么事情，只是会更新一下记录的修改时间，这是因为这个/tomcat-7/webapps/应用名目录下的文件，要么是jsp文件，要么是其他文件，而Tomcat只会管jsp文件，而对于jsp文件如果发生了修改，jsp自带的机制会处理修改的。

如果发生改变的是/tomcat-7/conf/Catalina/localhost/应用名.xml文件，那么就是先undeploy，然后再deploy，和热加载其实类似。对于undeploy就不多说了，就是讲当前应用从host从移除，这就包括了当前应用的停止和销毁，然后还会从已部署列表中移除当前应用，然后调用deployApps()就可以重新部署应用了。