# 1、tomcat 原理

1. 打破双亲委派

以Tomcat类加载为例，Tomcat 如果使用默认的双亲委派类加载机制行不行？

我们思考一下：Tomcat是个web容器， 那么它要解决什么问题：

1. 一个web容器可能需要部署两个应用程序，不同的应用程序可能会依赖同一个第三方类库的不同版本，不能要求同一个类库在同一个服务器只有一份，因此要保证每个应用程序的类库都是独立的，保证相互隔离。

2. 部署在同一个web容器中相同的类库相同的版本可以共享。否则，如果服务器有10个应用程序，那么要有10份相同的类库加载进虚拟机。

3. web容器也有自己依赖的类库，不能与应用程序的类库混淆。基于安全考虑，应该让容器的类库和程序的类库隔离开来。

4. web容器要支持jsp的修改，我们知道，jsp 文件最终也是要编译成class文件才能在虚拟机中运行，但程序运行后修改jsp已经是司空见惯的事情， web容器需要支持 jsp 修改后不用重启。

再看看我们的问题：Tomcat 如果使用默认的双亲委派类加载机制行不行？

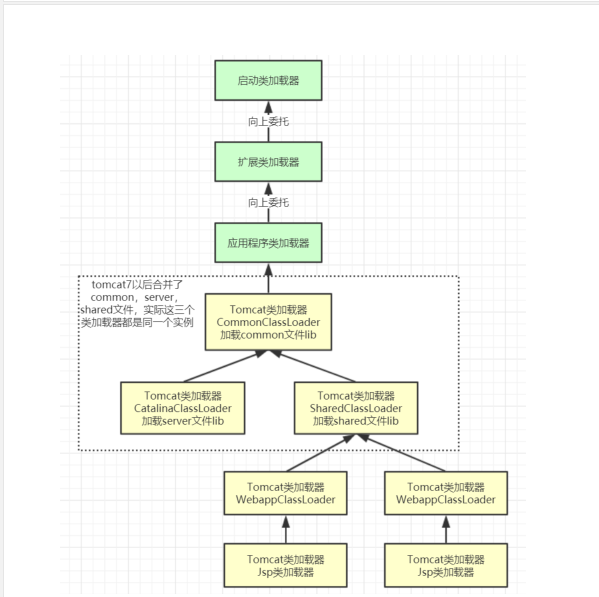
答案是不行的。为什么？

第一个问题，如果使用默认的类加载器机制，那么是无法加载两个相同类库的不同版本的，默认的类加器是不管你是什么版本的，只在乎你的全限定类名，并且只有一份。

第二个问题，默认的类加载器是能够实现的，因为他的职责就是保证唯一性。

第三个问题和第一个问题一样。

我们再看第四个问题，我们想我们要怎么实现jsp文件的热加载，jsp 文件其实也就是class文件，那么如果修改了，但类名还是一样，类加载器会直接取方法区中已经存在的，修改后的jsp是不会重新加载的。那么怎么办呢？我们可以直接卸载掉这jsp文件的类加载器，所以你应该想到了，每个jsp文件对应一个唯一的类加载器，当一个jsp文件修改了，就直接卸载这个jsp类加载器。重新创建类加载器，重新加载jsp文件。



tomcat的几个主要类加载器：

commonLoader：Tomcat最基本的类加载器，加载路径中的class可以被

Tomcat容器本身以及各个Webapp访问；

catalinaLoader：Tomcat容器私有的类加载器，加载路径中的class对于

Webapp不可见；

sharedLoader：各个Webapp共享的类加载器，加载路径中的class对于所有

Webapp可见，但是对于Tomcat容器不可见；

WebappClassLoader：各个Webapp私有的类加载器，加载路径中的class只对

当前Webapp可见；

从图中的委派关系中可以看出：

**CommonClassLoader**能加载的类都可以被CatalinaClassLoader和SharedClassLoader使用，从而实现了公有类库的共用，而CatalinaClassLoader和SharedClassLoader自己能加载的类则与对方相互隔离。

**WebAppClassLoader**可以使用SharedClassLoader加载到的类，但各个WebAppClassLoader实例之间相互隔离。

**JasperLoader**的加载范围仅仅是这个JSP文件所编译出来的那一个.Class文件，它出现的目的就是为了被丢弃：当Web容器检测到JSP文件被修改时，会替换掉目前的JasperLoader的实例，并通过再建立一个新的Jsp类加载器来实现JSP文件的热加载功能。

tomcat 这种类加载机制违背了java 推荐的双亲委派模型了吗？答案是：违背了。

我们前面说过，双亲委派机制要求除了顶层的启动类加载器之外，其余的类加载器都应当由自己的父类加载器加载。很显然，tomcat 不是这样实现，tomcat 为了实现隔离性，没有遵守这个约定，每个webappClassLoader加载自己的目录下的class文件，不会传递给父类加载器，打破了双亲委派机制。

Tomcat对于相同jar包不同版本的处理是通过不同的类加载器实现，每个app使用自己的类加载器，jvm中是允许相同名城相同路径的类存在的，只是类加载器不同即可区分

# 2、tomcat使用

1、禁用AJP（Apache JServer Protocol）

AJPv13协议是面向包的。WEB服务器和Servlet容器通过TCP连接来交互；为了节省SOCKET创建的昂贵代价，WEB服务器会尝试维护一个永久TCP连接到servlet容器，并且在多个请求和响应周期过程会重用连接。

我们一般是使用Nginx+Tomcat的架构，所以用不着AJP协议，把AJP连接器禁用。

执行器参数说明



<Executor name="tomcatThreadPool" namePrefix="catalina-exec-"

maxThreads="800" minSpareThreads="100" maxQueueSize="100" prestartminSpareThreads="true"/>

最大线程数默认是200，实际使用设置1000以内，600-800居多

|  |  |
| --- | --- |
| Attribute | Description |
| threadPriority （优先级） | (int) 执行程序中线程的线程优先级，默认值为 5（Thread.NORM\_PRIORITY常量的值） |
| daemon（守护进程） | (布尔) 线程是否应该是守护程序线程，默认值为 true |
| namePrefix（名称前缀） | (String) 执行程序创建的每个线程的名称前缀。单个线程的线程名称将为namePrefix+threadNumber |
| **maxThreads**（最大线程数） | (int) 此池中活动线程的最大数量，默认为 200 |
| **minSpareThreads**（最小活跃线程数） | (int) 始终保持活动状态的最小线程数（空闲和活动），默认值为 25 |
| maxIdleTime（空闲线程等待时间） | (int) 空闲线程关闭之前的毫秒数，除非活动线程的数目小于或等于minSpareThreads。默认值为60000（1分钟） |
| **maxQueueSize**（最大的等待队里数，超过则请求拒绝） | (int) 在我们拒绝执行之前可以排队等待执行的可运行任务的最大数量。默认值为Integer.MAX\_VALUE |
| **prestartminSpareThreads**（是否在启动时就生成minSpareThreads个线程） | (boolean) 在启动执行程序时是否应启动minSpareThreads，默认值为false |
| threadRenewalDelay（重建线程的时间间隔） | (long) 如果配置了ThreadLocalLeakPreventionListener，它将通知该执行程序已停止的上下文。上下文停止后，池中的线程将更新。为避免同时更新所有线程，此选项设置了任意两个线程之间的延迟。该值以毫秒为单位，默认值为1000ms。如果值为负，则不更新线程。 |

连接器参数说明

<Connector executor="tomcatThreadPool" port="8080" protocol="org.apache.coyote.http11.Http11Nio2Protocol"

connectionTimeout="20000" redirectPort="8443"

enableLookups="false" maxPostSize="10485760" URIEncoding="UTF-8" acceptCount="100" acceptorThreadCount="2" disableUploadTimeout="true" maxConnections="10000" SSLEnabled="false"/>

|  |  |
| --- | --- |
| Attribute | Description |
| allowTrace | 如果需要服务器能够处理用户的HAED/TRACE请求，这个值应该设置为true，默认值是false |
| asyncTimeout | 默认超不时候以毫秒为单位的异步恳求。若是没有指定，该属性被设置为10000（10秒） |
| **enableLookups** | 设置为true是否要调用以 request.getRemoteHost()执行DNS查找以返回远程客户端的实际主机名。设置为false跳过DNS查找并改为以字符串形式返回IP地址（从而提高性能）。默认情况下，DNS查找被禁用。 |
| maxHeaderCount | 容器允许的请求头字段的最大数目。请求中包含比指定的限制更多的头字段将被拒绝。值小于0表示没有限制。如果没有指定，默认设置为100。 |
| maxParameterCount | 将被容器自动解析的最大数量的参数和值对（GET加上POST）。参数值对超出此限制将被忽略。值小于0表示没有限制。如果没有指定，默认为10000。请注意， FailedRequestFilter 过滤器可以用来拒绝达到了极限值的请求。 |
| **maxPostSize** | 容器FORM URL参数解析将处理的POST的最大大小（以字节为单位）。可以通过将此属性设置为小于零的值来禁用该限制。如果未指定，则此属性设置为2097152（2兆字节）。请注意，FailedRequestFilter 可以使用拒绝超过此限制的请求。 |
| maxSavePostSize | 将被容器在FORM或CLIENT-CERT认证中保存/缓冲的POST的最大尺寸（以字节为单位）。对于这两种类型的身份验证，在用户身份验证之 前，POST将被保存/缓冲。对于POST CLIENT-CERT认证，处理该请求的SSL握手和缓冲清空期间，POST将被缓存。对于Form认证，POST将被保存，同时用户将被重定向到登陆 表单。POST将被一直保留直到用户成功认证或者认证请求关联的会话超时。将此属性设置为-1可以禁用此限制。将此属性设置为0，POST数据在身份验证 过程中将不被保存。如果没有指定，该属性设置为4096（4千字节）。 |
| parseBodyMethods | 以逗号分隔的HTTP方法列表，通过方法列表，等同于POST方法，request 正文将被解析成请求参数。这在RESTful应用程序要支持以POST式的语义解析PUT请求中是非常有用的。需要注意的是设置其他值（不是POST）会导致Tomcat的行为违反servlet规范的目的。在这里为了符合HTTP规范明确禁止HTTP方法TRACE。默认值是POST |
| **port** | 连接器 将在其上创建服务器套接字并等待传入连接的TCP端口号。您的操作系统将仅允许一个服务器应用程序侦听特定IP地址上的特定端口号。如果使用特殊值0（零），则Tomcat将随机选择一个空闲端口用于此连接器。这通常仅在嵌入式和测试应用程序中有用。 |
| **protocol** | 设置协议以处理传入流量。默认值为 HTTP/1.1使用自动切换机制选择基于Java NIO的连接器或基于APR / native的连接器。如果PATH（Windows）或LD\_LIBRARY\_PATH（在大多数Unix系统上）环境变量包含Tomcat本机库，并且AprLifecycleListener用于初始化APR的库的useAprConnector属性设置为 true，则将使用APR /本机连接器。如果找不到本机库或未配置属性，则将使用基于Java NIO的连接器。请注意，APR /本机连接器的HTTPS设置与Java连接器的设置不同。 |
| 要使用显式协议而不是依赖于上述自动切换机制，可以使用以下值：  org.apache.coyote.http11.Http11NioProtocol-非阻塞Java NIO连接器  org.apache.coyote.http11.Http11Nio2Protocol-非阻塞Java NIO2连接器-APR  org.apache.coyote.http11.Http11AprProtocol/本地连接器。  也可以使用自定义实现。  看看我们的连接器比较表。对于Java和Java连接器，http和https的配置相同。  有关APR连接器和特定于APR的SSL设置的更多信息，请访问APR文档 | |
| proxyName | 如果这个连接正在使用的代理服务器配置，配置该属性指定的服务器的名称，可以调用request.getServerName（）返回。有关更多信息，请参见代理支持。 |
| proxyPort | 如果这个连接正在使用的代理服务器配置，配置该属性指定服务器端口，可以调用request.getServerPort（）返回。有关更多信息，请参见代理支持。 |
| redirectPort | 如果该连接器支持非SSL请求，并且接收到的请求为满足安全约束需要SSL传输， Catalina 将自动将请求重定向到指定的端口号。 |
| scheme | 将该属性设置为你想调用request.getScheme（）返回的协议的名称。例如，对于SSL连接器，你会将此属性设置为“HTTPS ”。默认值是“ HTTP ”。 |
| secure | 如果你想调用request.isSecure（）收到此连接器的请求返回true，请该该属性设置为true。您希望SSL连接器或非SSL连接器接收数据通过一个SSL加速器，像加密卡，SSL设备，甚至一个web服务器。默认值是假的。 |
| **URIEncoding** | 解决我们的乱码问题，这将指定使用的字符编码，来解码URI字符。如果没有指定，ISO-8859-1将被使用。 |
| useBodyEncodingForURI | 这指定是否应该用于URI查询参数，而不是使用URIEncoding contentType中指定的编码。此设置兼容性Tomcat 4.1.x版（该版在contentType中指定编码，或者使用request.setCharacterEncoding的方法显式设置（参数为 URL传来的值）。默认值false。 |
| useIPVHosts | 将该属性设置为true会导致Tomcat使用收到请求的IP地址，来确定将请求发送到哪个主机。默认值是假的。 |
| xpoweredBy | 将此属性设置为true会导致Tomcat支持使用Servlet规范的通知，（在规范中推荐使用头字段）。默认值是假的。 |

标准的连接器（BIO，NIO和APR/native）都支持以下属性

|  |  |
| --- | --- |
| Attribute | Description |
| **acceptCount** | 当所有可能的请求处理线程都在使用时，传入连接请求的最大队列长度。当队列满时收到的任何请求将被拒绝。默认值是100。 |
| **acceptorThreadCount** | 用于接受连接的线程的数量。在一个多CPU的机器上，增加该值，虽然你可能不会真正需要超过2个。此外，有很多非保持活动连接，您可能需要增加这个值。默认值是 1。 |
| acceptorThreadPriority | 接收器线程的优先级。该线程用来接受新的连接。默认值是5（java.lang.Thread.NORM\_PRIORITY常量）。更多这个优先级是什么意思的详细信息，请查看java.lang.Thread的类的JavaDoc 。 |
| address | 对于拥有多个IP地址的服务器，该属性指定哪个地址将被用于在指定端口上监听。默认情况下，该端口将被用于与服务器相关联的所有IP地址。 |
| bindOnInit | 控制连接器绑定时套接字的使用。缺省情况，当连接器被启动时套接字被绑定和当连接器被销毁时套接字解除绑定。如果设置为false，连接器启动时套接字被绑定，连接器停止时套接字解除绑定。 |
| compressableMimeType | 该值是一个被用于HTTP压缩的逗号分隔的MIME类型列表。默认值是text / html类型，为text / xml，text / plain。 |
| **compression** | 通常会在ngnix里面配置压缩 ，开启压缩GZIP 为了节省服务器带宽，连接器可以使用HTTP/1.1 GZIP压缩。可接受的参数的值是“off ”（禁用压缩），“on ”（允许压缩，这会导致文本数据被压缩），“force ”（强制在所有的情况下压缩），或者一个整数值（这是相当于为“on”，但指定了输出之前被压缩的数据最小量）。如果不知道内容长度但被设置为“on”或更积极的压缩，输出的数据也将被压缩。如果没有指定，该属性被设置为“关”。 注意：这是使用压缩（节省您的带宽）和使用sendfile功能（节省你的CPU周期）之间的权衡。如果连接器支持sendfile功能，例如NIO连接，则使用sendfile将优先于压缩。症状是48 KB的静态文件将未压缩就发送。你可以如下文所述通过设置连接器的useSendfile属性来关闭sendfile，或在默认的conf/web.xml或者你的web应用的web.xml中配置DefaultServlet来改变sendfile的使用量阈值。 |
| compressionMinSize | 如果压缩被设置为“on”，那么该属性可以用于指定在输出之前被压缩的数据的最小量。如果未指定，此属性默认为“2048”。 |
| connectionLinger | 连接器的套接字被关闭时的逗留秒数。如果没有指定，将使用默认的JVM。 |
| connectionTimeout | 在将提交的请求URI行呈现之后，连接器将等待接受连接的毫秒数。使用值-1表示没有超时（即无限）。默认值是60000（60秒），但请注意，Tomcat的标准server.xml中，设置为20000（即20秒）。 |
| **connectionUploadTimeout** | 上传数据过程中，指定的以毫秒为单位超时时间。只有在设置disableUploadTimeout为false有效。 |
| **disableUploadTimeout** | 此标志允许servlet容器在数据上传时使用不同的连接超时，通常较长。如果没有指定，该属性被设置为true，禁用上传超时。 |
| **executor** | 指向Executor元素的引用。如果这个属性被设置，并且被命名的executor存在，连接器将使用这个executor，而其他所有线程相关属性将被忽略。请注意共享的executor如果没有指定到一个连接器，则该连接器将使用一个私有的，内部的executor来提供线程池。 |
| executorTerminationTimeoutMillis | The time that the private internal executor will wait for request processing threads to terminate before continuing with the process of stopping the connector. If not set, the default is 0 (zero) for the BIO connector and 5000 (5 seconds) for the NIO and APR/native connectors. |
| keepAliveTimeout | 此连接器在关闭连接之前将等待另一个HTTP请求的毫秒数。默认值是使用已设置的connectionTimeout属性的值。使用值-1表示没有超时（即无限）。 |
| **maxConnections** | 在任何给定的时间服务器接受并处理的最大连接数。当这个数字已经达到了，服务器将不会接受任何连接，直到连接的数量降到低于此值。基于acceptCount的设置，操作系统可能仍然接受连接。默认值根据不同的连接器类型而不同。对于BIO，默认的是maxThreads的值，除非使用了Executor，在这种情况下默认值是executor的maxThreads值 。对于NIO的默认值是10000。APR /native的默认值是8192。 需要注意的是Windows系统的APR/native，所配置的值将减少到小于或等于maxConnections的1024的倍数的最大值。这样做是出于性能方面的考虑。如果设置的值-1，maxConnections功能被禁用，而且连接数将不做计算。 |
| maxExtensionSize | 限制HTTP区块请求中区块扩展的总长度。如果值为-1，则不施加任何限制。如果未指定，8192将使用默认值。 |
| maxHttpHeaderSize | 请求和响应的HTTP头的（以字节为单位的）最大尺寸。如果没有指定，该属性被设置为8192（8 KB）。 |
| maxKeepAliveRequests | HTTP请求最大长连接个数。将此属性设置为1，将禁用HTTP/1.0、以及HTTP/1.1的长连接。设置为-1，不禁用。如果没有指定，该属性被设置为100。 |
| maxSwallowSize | Tomcat会为中止的上载而吞下的请求正文字节的最大数量（不包括传输编码开销）。上载中止是指Tomcat知道将忽略请求主体，但客户端仍将其发送。如果Tomcat不吞咽该主体，则客户端不太可能看到响应。如果未指定，将使用默认值2097152（2兆字节）。小于零的值表示不应强制执行任何限制。 |
| **maxThreads** | 最多同时处理的连接数，Tomcat使用线程来处理接收的每个请求。这个值表示Tomcat可创建的最大的线程数。如果没有指定，该属性被设置为200。如果使用了execute将忽略此连接器的该属性，连接器将使用execute，而不是一个内部线程池来处理请求。 |
| maxTrailerSize | 限制一个分块的HTTP请求中的最后一个块的尾随标头的总长度。如果该值是-1，没有限制的被强加。如果没有指定，默认值是8192。 |
| **minSpareThreads** | 始终保持运行最小线程数。如果没有指定，则默认为10。 |
| noCompressionUserAgents | 该值是一个正则表达式（使用java.util.regex），匹配不应该使用压缩的HTTP客户端的用户代理标头。因为这些客户端，虽然他们宣称支持压缩功能，但实现不完整。默认值是一个空字符串（正则表达式匹配禁用）。 |
| processorCache | 协议处理器缓存Processor对象以提高性能。此设置规定了这些对象有多少能得到缓存。-1意味着无限制，默认为200。如果不使用Servlet 3.0的异步处理，一个好的默认是使用maxThreads设置。如果使用Servlet 3.0的异步处理，一个好的默认是使用maxThreads和最大预期的并发请求（同步和异步）的最大值中的较大值。 |
| restrictedUserAgents | 该值是一个正则表达式（使用java.util.regex），匹配用户代理头的HTTP浏览器将不能使用HTTP/1.1或HTTP/1.0长连接，即使该浏览器宣称支持这些功能的。默认值是一个空字符串（正则表达式匹配禁用）。 |
| server | 覆盖服务器的HTTP响应头。如果设置了这个属性的值将覆盖Web应用程序设置的Tomcat的默认头和任何服务器头。如果没有设置，应用程序指定的任何值将被使用。如果应用程序没有指定一个值，那么Apache-Coyote/1.1将被使用。除非你是偏执狂，你将不再需要此功能。 |
| socketBuffer | 为套接字输出缓冲而提供的缓冲区的大小（以字节为单位）。-1可以被指定来禁止使用的缓冲区。默认情况下，一个9000个字节的缓冲区将被使用。 |
| **SSLEnabled** | 在连接器上使用此属性来启用SSL加密传输。如果要打开SSL握手/加密/解密，请设置true。默认值是false。当设置这个值为true时，为了传递正确的request.getScheme（）和 request.isSecure（）到servlets，你需要设置scheme和secure属性。更多信息请查看SSL支持。 |
| tcpNoDelay | 如果设置为true，TCP\_NO\_DELAY选项将被设置在服务器上的套接字上，在大多数情况下，这样可以提高性能。默认设置为true。 |
| threadPriority | 在JVM中请求处理线程的优先级。默认值是5（java.lang.Thread.NORM\_PRIORITY常量值）。关于优先级的更多详细信息，请查看java.lang.Thread的类的JavaDoc |
| upgradeAsyncWriteBufferSize | The default size of the buffer to allocate to for asynchronous writes that can not be completed in a single operation. Data that can’t be written immediately will be stored in this buffer until it can be written. If more data needs to be stored than space is available in the buffer than the size of the buffer will be increased for the duration of the write. If not specified the default value of 8192 will be used. |