1. 你怎样给 tomcat 去调优?

1. JVM 参数调优：-Xms<size> 表示 JVM 初始化堆的大小，-Xmx<size>表示 JVM 堆的最大值。这两个值的大小一般根据需要进行设置。当应用程序需要的内存超出堆的最大值时虚拟机就会提示内存溢出，并且导致应用服务崩溃。因此一般建议堆的最大值设置为可用内存的最大值的 80%。在 catalina.bat 中，设置 JAVA_OPTS='-Xms256m -Xmx512m'，表示初始化内存为 256MB，可以使用的最大内存为 512MB。

2. 禁用 DNS 查询

当 web 应用程序向要记录客户端的信息时，它也会记录客户端的 IP 地址或者通过域名服务器查找机器名转换为 IP 地址。DNS 查询需要占用网络，并且包括可能从很多很远的服务器或者不起作用的服务器上去获取对应的 IP 的过程，这样会消耗一定的时间。为了消除 DNS 查询对性能的影响我们可以关闭 DNS 查询，方式是修改 server.xml 文件中的 enableLookups 参数值：

Tomcat4

<Connector className="org.apache.coyote.tomcat4.CoyoteConnector" port="80" minProcessors="5" maxProcessors="75" enableLookups="false" redirectPort="8443" acceptCount="100" debug="0" connectionTimeout="20000" useURIValidationHack="false" disableUploadTimeout="true" />

Tomcat5

<Connector port="80" maxThreads="150" minSpareThreads="25"

maxSpareThreads="75" enableLookups="false" redirectPort="8443"

acceptCount="100" debug="0" connectionTimeout="20000"

disableUploadTimeout="true"/>

3. 调整线程数

通过应用程序的连接器（Connector）进行性能控制的的参数是创建的处理请求的线程数。Tomcat 使用线程池加速响应速度来处理请求。在 Java 中线程是程序运行时的路径，是在一个程序中与其它控制线程无关的、能够独立运行的代码段。它们共享相同的地址空间。多线程帮助程序员写出 CPU 最大利用率的高效程序，使空闲时间保持最低，从而接受更多的请求。

Tomcat4 中可以通过修改 minProcessors 和 maxProcessors 的值来控制线程数。这些值在安装后就已经设定为默认值并且是足够使用的，但是随着站点的扩容而改大这些值。minProcessors 服务器启动时创建的处理请求的线程数应该足够处理一个小量的负载。也就是说，如果一天内每秒仅发生 5 次单击事件，并且每个请求任务处理需要 1 秒钟，那么预先设置线程数为 5 就足够了。但在你的站点访问量较大时就需要设置更大的线程数，指定为参数 maxProcessors 的值。maxProcessors 的值也是有上限的，应防止流量不可控制（或者恶意的服务攻击），从而导致超出了虚拟机使用内存的大小。如果要加大并发连接数，应同时加大这两个参数。web server 允许的最大连接数还受制于操作系统的内核参数设置，通常 Windows 是 2000 个左右，Linux 是 1000 个左右。

在 Tomcat5 对这些参数进行了调整，请看下面属性：

maxThreads　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。

acceptCount　指定当所有可以使用的处理请求的线程数都被使用时，可以放到处理队列中的

请求数，超过这个数的请求将不予处理。

connnectionTimeout　网络连接超时，单位：毫秒。设置为 0 表示永不超时，这样设置有隐患的。通常可设置为 30000 毫秒。

minSpareThreads　Tomcat 初始化时创建的线程数。

maxSpareThreads　一旦创建的线程超过这个值，Tomcat 就会关闭不再需要的 socket 线程。

最好的方式是多设置几次并且进行测试，观察响应时间和内存使用情况。在不同的机器、操作系统或虚拟机组合的情况下可能会不同，而且并不是所有人的 web 站点的流量都是一样的，因此没有一刀切的方案来确定线程数的值。

2. 如何加大 tomcat 连接数

在 tomcat 配置文件 server.xml 中的<Connector />配置中，和连接数相关的参数有：

minProcessors：最小空闲连接线程数，用于提高系统处理性能，默认值为 10

maxProcessors：最大连接线程数，即：并发处理的最大请求数，默认值为 75

acceptCount：允许的最大连接数，应大于等于 maxProcessors，默认值为 100

enableLookups：是否反查域名，取值为：true 或 false。为了提高处理能力，应设置为 false

connectionTimeout：网络连接超时，单位：毫秒。设置为 0 表示永不超时，这样设置有隐患的。通常可设置为 30000 毫秒。

其中和最大连接数相关的参数为 maxProcessors 和 acceptCount。如果要加大并发连接数，应同时加大这两个参数。

web server 允许的最大连接数还受制于操作系统的内核参数设置，通常 Windows 是 2000 个左右，Linux 是 1000 个左右。tomcat5 中的配置示例：

```
<Connector port="8080"
        maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
        enableLookups="false" redirectPort="8443" acceptCount="100"
        debug="0" connectionTimeout="20000"
        disableUploadTimeout="true" />
```

对于其他端口的侦听配置，以此类推。

3. tomcat 中如何禁止列目录下的文件

在{tomcat_home}/conf/web.xml 中，把 listings 参数设置成 false 即可，如下：

```
<init-param>
<param-name>listings</param-name>
<param-value>false</param-value>
</init-param>
<init-param>
<param-name>listings</param-name>
```

<param-value>false</param-value>

</init-param>

4.怎样加大 tomcat 的内存。

首先检查程序有没有限入死循环

这个问题主要还是由这个问题 java.lang.OutOfMemoryError: Java heap space 引起的。第一次出现这样的的问题以后，引发了其他的问题。在网上一查可能是 JAVA 的堆栈设置太小的原因。
跟据网上的答案大致有这两种解决方法：
1、设置环境变量

解决方法：手动设置 Heap size
修改 TOMCAT_HOME/bin/catalina.sh
set JAVA_OPTS= -Xms32m -Xmx512m
可以根据自己机器的内存进行更改。

2、java -Xms32m -Xmx800m className
就是在执行 JAVA 类文件时加上这个参数，其中 className 是需要执行的确类名。（包括包名）
这个解决问题了。而且执行的速度比没有设置的时候快很多。

如果在测试的时候可能会用 Eclispe 这时候就需要在 Eclipse ->run -arguments 中的 VM arguments 中输入-Xms32m -Xmx800m 这个参数就可以了。

后来在 Eclilpse 中修改了启动参数，在 VM arguments 加入了-Xms32m -Xmx800m，问题解决。

一、java.lang.OutOfMemoryError: PermGen space

PermGen space 的全称是 Permanent Generation space,是指内存的永久保存区域,
这块内存主要是被 JVM 存放 Class 和 Meta 信息的,Class 在被 Loader 时就会被放到 PermGen space 中,
它和存放类实例(Instance)的 Heap 区域不同,GC(Garbage Collection)不会在主程序运行期对
PermGen space 进行清理，所以如果你的应用中有很多 CLASS 的话,就很可能出现 PermGen space 错误,
这种错误常见在 web 服务器对 JSP 进行 pre compile 的时候。如果你的 WEB APP 下都用了大量的第三方 jar, 其大小
超过了 jvm 默认的大小(4M)那么就会产生此错误信息了。
解决方法： 手动设置 MaxPermSize 大小

修改 TOMCAT_HOME/bin/catalina.sh
在“echo "Using CATALINA_BASE: $CATALINA_BASE"”上面加入以下行：
JAVA_OPTS="-server -XX:PermSize=64M -XX:MaxPermSize=128m
建议：将相同的第三方 jar 文件移置到 tomcat/shared/lib 目录下，这样可以达到减少 jar 文档重复占用内存的目的。

二、java.lang.OutOfMemoryError: Java heap space

Heap size 设置

JVM 堆的设置是指 java 程序运行过程中 JVM 可以调配使用的内存空间的设置.JVM 在启动的时候会自动设置 Heap size 的值，

其初始空间(即-Xms)是物理内存的 1/64，最大空间(-Xmx)是物理内存的 1/4。可以利用 JVM 提供的-Xmn -Xms -Xmx 等选项可

进行设置。Heap size 的大小是 Young Generation 和 Tenured Generaion 之和。

提示：在 JVM 中如果 98％的时间是用于 GC 且可用的 Heap size 不足 2％的时候将抛出此异常信息。

提示：Heap Size 最大不要超过可用物理内存的 80％，一般的要将-Xms 和-Xmx 选项设置为相同，而-Xmn 为 1/4 的-Xmx 值。

解决方法：手动设置 Heap size

修改 TOMCAT_HOME/bin/catalina.sh

在“echo "Using CATALINA_BASE: $CATALINA_BASE"”上面加入以下行：

JAVA_OPTS="-server -Xms800m -Xmx800m -XX:MaxNewSize=256m"

三、实例，以下给出 1G 内存环境下 java jvm 的参数设置参考：

JAVA_OPTS="-server -Xms800m -Xmx800m -XX:PermSize=64M -XX:MaxNewSize=256m -XX:MaxPermSize=128m -Djava.awt.headless=true "

很大的 web 工程，用 tomcat 默认分配的内存空间无法启动，如果不是在 myeclipse 中启动 tomcat 可以对 tomcat 这样设置：

TOMCAT_HOME/bin/catalina.bat 中添加这样一句话：

set JAVA_OPTS=-server -Xms2048m -Xmx4096m -XX:PermSize=512M -XX:MaxPermSize=1024M -Duser.timezone=GMT+08

或者

set JAVA_OPTS= -Xmx1024M -Xms512M -XX:MaxPermSize=256m

如果要在 myeclipse 中启动，上述的修改就不起作用了，可如下设置：

Myeclipse->preferences->myeclipse->servers->tomcat->tomcat×.×->JDK 面板中的

Optional Java VM arguments 中添加：-Xmx1024M -Xms512M -XX:MaxPermSize=256m

以上是转贴，但本人遇见的问题是：在 myeclipse 中启动 Tomcat 时，提示"ava.lang.OutOfMemoryError: Java heap space"，解决办法就是：

Myeclipse->preferences->myeclipse->servers->tomcat->tomcat×.×->JDK 面板中的

Optional Java VM arguments 中添加：-Xmx1024M -Xms512M -XX:MaxPermSize=256m

5.Tomcat 有几种部署方式

tomcat 中四种部署项目的方法

第一种方法：

在 tomcat 中的 conf 目录中，在 server.xml 中的，<host/>节点中添加：

<Context path="/hello"

docBase="D:/eclipse3.2.2/forwebtoolsworkspacehello/WebRoot" debug="0"

privileged="true">

</Context>

至于 Context 节点属性，可详细见相关文档。

第二种方法：

将 web 项目文件件拷贝到 webapps 目录中。

第三种方法：

很灵活，在 conf 目录中，新建 Catalina（注意大小写）\ localhost 目录，在该目录中新建一个 xml 文件，名字可以随意取，只要和当前文件中的文件名不重复就行了，该 xml 文件的内容为：

<Context path="/hello" docBase="D:eclipse3.2.2forwebtoolsworkspacehelloWebRoot"

debug="0" privileged="true">

</Context>

第 3 个方法有个优点，可以定义别名。服务器端运行的项目名称为 path，外部访问的 URL 则使用 XML 的文件名。这个方法很方便的隐藏了项目的名称，对一些项目名称被固定不能更换，但外部访问时又想换个路径，非常有效。

第 2、3 还有优点，可以定义一些个性配置，如数据源的配置等。

第四种办法,:

可以用 tomcat 在线后台管理器,一般 tomcat 都打开了,直接上传 war 就可以

6.Tomcat 的优化经验。

Tomcat 作为 Web 服务器，它的处理性能直接关系到用户体验，下面是几种常见的优化措施:

- **去掉对 web.xml 的监视，把 jsp 提前编辑成 Servlet。有富余物理内存的情况，加大 tomcat 使用的 jvm 的内存。**
- **服务器资源**

  服务器所能提供 CPU、内存、硬盘的性能对处理能力有决定性影响。

  - 对于高并发情况下会有大量的运算，那么 CPU 的速度会直接影响到处理速度。
  - 内存在大量数据处理的情况下，将会有较大的内存容量需求，可以用 -Xmx -Xms -XX:MaxPermSize 等参数对内存不同功能块进行划分。我们之前就遇到过内存分配不足，导致虚拟机一直处于 full GC，从而导致处理能力严重下降。
  - 硬盘主要问题就是读写性能，当大量文件进行读写时，磁盘极容易成为性能瓶颈。最好的办法还是利用下面提到的缓存。
- **利用缓存和压缩**

  对于静态页面最好是能够缓存起来，这样就不必每次从磁盘上读。这里我们采用了 Nginx 作为缓存服务器，将图片、css、js 文件都进行了缓存，有效的减少了后端 tomcat 的访问。　　另外，为了能加快网络传输速度，开启

gzip 压缩也是必不可少的。但考虑到 tomcat 已经需要处理很多东西了，所以把这个压缩的工作就交给前端的 Nginx 来完成。 除了文本可以用 gzip 压缩，其实很多图片也可以用图像处理工具预先进行压缩，找到一个平衡点可以让画质损失很小而文件可以减小很多。曾经我就见过一个图片从 300 多 kb 压缩到几十 kb，自己几乎看不出来区别。

- **采用集群**

  单个服务器性能总是有限的，最好的办法自然是实现横向扩展，那么组建 tomcat 集群是有效提升性能的手段。我们还是采用了 Nginx 来作为请求分流的服务器，后端多个 tomcat 共享 session 来协同工作。可以参考之前写的《利用 nginx+tomcat+memcached 组建 web 服务器负载均衡》。

- **优化 tomcat 参数**

这里以 tomcat7 的参数配置为例，需要修改 conf/server.xml 文件，主要是优化连接配置，关闭客户端 dns 查询。

```
<Connector port="8080"
   protocol="org.apache.coyote.http11.Http11NioProtocol"
   connectionTimeout="20000"
   redirectPort="8443"
   maxThreads="500"
   minSpareThreads="20"
   acceptCount="100"
   disableUploadTimeout="true"
   enableLookups="false"
   URIEncoding="UTF-8" />
```