提升J 页面响应速度的七大秘籍绝招 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/252/2021_2022__E6_8F_90_E5_8D_87J__E9_A1_c104_252414.htm 方法一：在Servlet的init()方法中缓存数据 当应用服务器初始化servlet实例之后，为客户端请求提供服务之前，它会调用这个servlet的init()方法。在一个Servlet的生命周期中，init()方法只会被调用一次。通过在init()方法中缓存一些静态的数据或完成一些只需要执行一次的、耗时的操作，就可大大地提高系统性能。 例如，通过在init()方法中建立一个JDBC连接池是一个最佳例子，假设我们是用jdbc2.0的DataSource接口来取得数据库连接，在通常的情况下，我们需要通过JNDI来取得具体的数据源。我们可以想象在一个具体的应用中，如果每次SQL请求都要执行一次JNDI查询的话，那系统性能将会急剧下降。解决方法是如下代码，它通过缓存DataSource，使得下一次SQL调用时仍然可以继续利用它：以下是引用片段：public class ControllerServlet extends HttpServlet{ private Javax.sql.DataSource testDS = null. public void init(ServletConfig config) throws ServletException { super.init(config). Context ctx = null. try{ ctx = new InitialContext(). testDS = (Javax.sql.DataSource)ctx.lookup("jdbc/testDS"). }catch(NamingException ne){ne.printStackTrace().} }catch(Exception e){e.printStackTrace().} } public Javax.sql.DataSource getTestDS(){ return testDS. } ... ... } 方法 2:禁止Servlet和JSP 自动重载(auto-reloading) Servlet/JSP提供了一个实用的技术，即自动重载技术，它为开发人员提供了一个好

的开发环境，当你改变Servlet和JSP页面后而不必重启应用服务器。然而，这种技术在产品运行阶段对系统的资源是一个极大的损耗，因为它会给JSP引擎的类装载器(classloader)带来极大的负担。因此关闭自动重载功能对系统性能的提升是一个极大的帮助。 方法 3: 不要滥用Httpsession 在很多应用中，我们的程序需要保持客户端的状态，以便页面之间可以相互联系。但不幸的是由于HTTP具有天生无状态性，从而无法保存客户端的状态。因此一般的应用服务器都提供了session来保存客户的状态。在JSP应用服务器中，是通过HttpSession对像来实现session的功能的，但在方便的同时，它也给系统带来了不小的负担。因为每当你获得或更新session时，系统者要对它进行费时的序列化操作。你可以通过对Httpsession的以下几种处理方式来提升系统的性能。 如果没有必要，就应该关闭JSP页面中对HttpSession的缺省设置。 如果你没有明确指定的话，每个JSP页面都会缺省地创建一个HttpSession。如果你的JSP中不需要使用session的话，那可以通过如下的JSP页面指示符来禁止它：以下是引用片段：＜%@ page session="false"%＞ 不要在HttpSession中存放大的数据对像：如果你在HttpSession中存放大的数据对像的话，每当对它进行读写时，应用服务器都将对其进行序列化，从而增加了系统的额外负担。你在Httpsession中存放的数据对像越大，那系统的性能就下降得越快。 当你不需要HttpSession时，尽快地释放它：当你不再需要session时，你可以通过调用HttpSession.invalidate()方法来释放它。尽量将session的超时时间设得短一点：在JSP应用服务器中，有一个缺省的session的超时时间。当客户在这个时间之后没有进行任何操作的话

，系统会将相关的session自动从内存中释放。超时时间设得越大，系统的性能就会越低，因此最好的方法就是尽量使得它的值保持在一个较低的水平。 方法 4: 将页面输出进行压缩 压缩是解决数据冗余的一个好的方法，特别是在网络带宽不够发达的今天。有的浏览器支持gzip(GNU zip)进行来对HTML文件进行压缩，这种方法可以戏剧性地减少HTML文件的下载时间。因此，如果你将Servlet或JSP页面生成的HTML页面进行压缩的话，那用户就会觉得页面浏览速度会非常快。但不幸的是，不是所有的浏览器都支持gzip压缩，但你可以通过在你的程序中检查客户的浏览器是否支持它。下面就是关于这种方法实现的一个代码片段：以下是引用片段：public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException { OutputStream out = null. String encoding = request.getHeader("Accept-Encoding"). if (encoding != null amp. encoding.indexOf("gzip") != -1){ request.setHeader("Content-Encoding" , "gzip"). out = new GZIPOutputStream(request.getOutputStream()). } else if (encoding != null amp. encoding.indexOf("comdivss") != -1){ request.setHeader("Content-Encoding" , "comdivss"). out = new ZIPOutputStream(request.getOutputStream()). }else{ out = request.getOutputStream(). } ... ... } 方法 5: 使用线程池 应用服务器缺省地为每个不同的客户端请求创建一个线程进行处理，并为它们分派service()方法，当service()方法调用完成后，与之相应的线程也随之撤消。由于创建和撤消线程会耗费一定的系统资源，这种缺省模式降低了系统的性能。但所幸的是

我们可以通过创建一个线程池来改变这种状况。 另外，我们还要为这个线程池设置一个最小线程数和一个最大线程数。在应用服务器启动时，它会创建数量等于最小线程数的一个线程池，当客户有请求时，相应地从池从取出一个线程来进行处理，当处理完成后，再将线程重新放入到池中。如果池中的线程不够地话，系统会自动地增加池中线程的数量，但总量不能超过最大线程数。通过使用线程池，当客户端请求急剧增加时，系统的负载就会呈现的平滑的上升曲线，从而提高的系统的可伸缩性。 方法 6: 选择正确的页面包含机制 在JSP中有两种方法可以用来包含另一个页面： 1、使用include指示符以下是引用片段：＜%@ includee file= ”test.JSP ” %＞ 2、使用JSP指示符以下是引用片段：＜jsp:includee page= ”test.JSP ” flush= ”true ”/＞ 在实际中发现，如果使用第一种方法的话，可以使得系统性能更高。 方法 7:正确地确定Javabean的生命周期 JSP的一个强大的地方就是对JavaBean的支持。通过在JSP页面中使用JSP:useBean标签，可以将Javabean直接插入到一个JSP页面中。它的使用方法如下：以下是引用片段：＜JSP:useBean id="name" scope="page|request|session|application" class="package.className" type="typeName"＞ ＜/JSP:useBean＞ 其中scope属性指出了这个bean的生命周期。缺省的生命周期为page。如果你没有正确地选择bean的生命周期的话，它将影响系统的性能。 举例来说，如果你只想在一次请求中使用某个bean，但你却将这个bean的生命周期设置成了session，那当这次请求结束后，这个bean将仍然保留在内存中，除非session超时或用户关闭浏览器。这样会耗费一定的内存，

并无谓的增加了JVM垃圾收集器的工作量。因此为bean设置正确的生命周期，并在bean的使命结束后尽快地清理它们，会使用系统性能有一个提高。 其它一些有用的方法 1、在字符串连接操作中尽量不使用“ ”操作符：在Java编程中，我们常常使用“ ”操作符来将几个字符串连接起来，但你或许从来没有想到过它居然会对系统性能造成影响吧?由于字符串是常量，因此JVM会产生一些临时的对像。你使用的“ ”越多，生成的临时对像就越多，这样也会给系统性能带来一些影响。解决的方法是用StringBuffer对像来代替“ ”操作符。 2、避免使用System.out.println()方法：由于System.out.println()是一种同步调用，即在调用它时，磁盘I/O操作必须等待它的完成，因此我们要尽量避免对它的调用。但我们在调试程序时它又是一个必不可少的方便工具，为了解决这个矛盾，我建议你最好使用Log4j工具(http://Jakarta.apache.org)，它既可以方便调试，而不会产生System.out.println()这样的方法。 3、ServletOutputStream 与 PrintWriter的权衡：使用PrintWriter可能会带来一些小的开销，因为它将所有的原始输出都转换为字符流来输出，因此如果使用它来作为页面输出的话，系统要负担一个转换过程。而使用ServletOutputStream作为页面输出的话就不存在一个问题，但它是以二进制进行输出的。因此在实际应用中要权衡两者的利弊。 总结 本文的目的是通过对Servlet和JSP的一些调优技术来极大地提高你的应用程序的性能，并因此提升整个J2EE应用的性能。通过这些调优技术，你可以发现其实并不是某种技术平台(比如J2EE和.NET之争)决定了你的应用程序的性能，重要是你要对这种平台有一个较为深入的了解，这样你才能从根本上对自己的应用程序

做一个优化。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com