

专注于音视频解决方案：
视频会议（纯软、软硬结合）
在线教育（一体化解决方案）
企业培训直播
在线培训
在线路演
在线医疗等
QQ：1215972660

昵称：zero516cn
园龄：7年4个月
粉丝：266
关注：4
+加关注

<	2012年9月						>
日	一	二	三	四	五	六	
26	27	28	29	30	31	1	
2	3	4	5	6	7	8	
9	10	11	12	13	14	15	
16	17	18	19	20	21	22	
23	24	25	26	27	28	29	
30	1	2	3	4	5	6	

搜索

找找看

谷歌搜索

- 常用链接
- [我的随笔](#)
[我的评论](#)
[我的参与](#)
[最新评论](#)
[我的标签](#)

- 我的标签
- [Java\(9\)](#)
[Java源代码分析\(5\)](#)
[Java笔试、面试题\(4\)](#)
[C++\(3\)](#)
[算法\(3\)](#)
[线程\(3\)](#)
[Java虚拟机\(3\)](#)
[线程安全\(2\)](#)
[Oracle\(2\)](#)
[排序算法\(2\)](#)
[更多](#)

- 随笔分类 (88)
- [Ajax](#)
[C\(1\)](#)
[C++\(4\)](#)
[Hadoop\(8\)](#)
[HBase\(1\)](#)
[IBM FileNet ECM\(6\)](#)
[IBM ILOG\(5\)](#)
[Java\(27\)](#)
[JavaScript\(3\)](#)
[Jboss\(1\)](#)

深入研究Servlet线程安全性问题

本文参考链接（略加改动）：<http://www.yesky.com/334/1951334.shtml>

摘要：介绍了Servlet多线程机制，通过一个实例并结合Java 的内存模型说明引起Servlet线程不安全的原因，给出了保证Servlet线程安全的三种解决方案，并说明三种方案在实际开发中的取舍。

Servlet/JSP技术和ASP、PHP等相比，由于其多线程运行而具有很高的执行效率。由于Servlet/JSP默认是以多线程模式执行的，所以，在编写代码时需要非常细致地考虑多线程的安全性问题。然而，很多人编写Servlet/JSP程序时并没有注意到多线程安全性的问题，这往往造成编写的程序在少量用户访问时没有任何问题，而在并发用户上升到一定值时，就会经常出现一些莫名其妙的问题。

Servlet的多线程机制

Servlet体系结构是建立在Java多线程机制之上的，它的生命周期是由Web容器负责的。当客户端第一次请求某个Servlet时，Servlet容器将会根据web.xml配置文件实例化这个Servlet类。当有新的客户端请求该Servlet时，一般不会再实例化该Servlet类，也就是有多个线程在使用这个实例。Servlet容器会自动使用线程池等技术来支持系统的运行，如图1所示。

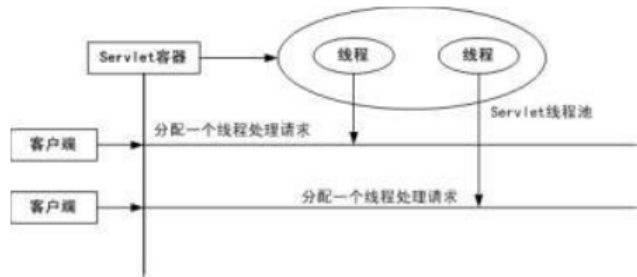


图 1 Servlet 线程池

这样，当两个或多个线程同时访问同一个Servlet时，可能会发生多个线程同时访问同一资源的情况，数据可能会变得不一致。所以在用Servlet构建的Web应用时如果不注意线程安全的问题，会使所写的Servlet程序有难以发现的错误。

Servlet的线程安全问题

Servlet的线程安全问题主要是由于实例变量使用不当而引起的，这里以一个现实的例子来说明。

```
1 public class ConcurrentTest extends HttpServlet {
2     PrintWriter output;
3     @Override
4     protected void service(HttpServletRequest request, HttpServletResponse response)
5         throws ServletException, IOException {
6         String username;
7         response.setContentType("text/html;charset=gb2312");
8         username=request.getParameter("username");
9         output=response.getWriter();
10        try {
11            //为了突出并发问题，在这设置一个延时
12            Thread.sleep(5000);
13            output.println("用户名:"+username+"<BR>");
14        } catch (Exception e) {
15            e.printStackTrace();
16        }
17    }
18 }
```

Linux(3)
Mina(2)
Oracle(9)
Servlet(1)
Scoop
Struts2(3)
Zookeeper
错误集锦(1)
设计模式(2)
数据结构(2)
算法(3)
系统架构(1)
线程安全(3)
音视频(1)
银行业务(1)

随笔档案 (123)

2016年5月(1)
2015年7月(11)
2015年6月(5)
2015年4月(1)
2015年3月(2)
2015年2月(1)
2015年1月(3)
2014年12月(3)
2014年11月(15)
2014年10月(6)
2014年7月(1)
2014年5月(2)
2014年4月(13)
2014年3月(12)
2014年2月(4)
2012年10月(27)
2012年9月(16)

最新评论

1. Re:Hbase split的三种方式和split的过程

HBase处理写请求时，不是先写WAL日志，再写memstore么？

--小北觅

2. Re:深入研究Servlet线程安全性问题
老哥，Tomcat中的
ApplicationFilterChain貌似是每个Url
Pattern的单例，想问下它是如何保证其
成员变量（例如pos与n）的线程安全
呢？这里我一直不理解

--wanxu

3. Re:C++引用详解

@ 青儿哥哥兄弟,这个是数组指针....

--AllenHuai

4. Re:Java运算符优先级

你好，我在别的博客里面看到的按位与、
按位或、按位异或 是优先级这样的&
、 ^ 、| ，请问这三个的优先级是
怎么样的？

--南墙、

5. Re:Java垃圾收集器

完全是copy《深入理解java虚拟机》的
内容，为了凑个博客数量，博主也是拼
了。

--windfallsheng

```
17     }  
18 }
```

该Servlet中定义了一个实例变量output，在service方法将其赋值为用户的输出。当一个用户访问该Servlet时，程序会正常的运行，但当多个用户并发访问时，就可能会出现其它用户的信息显示在另外一些用户的浏览器上的问题。这是一个严重的问题。为了突出并发问题，便于测试、观察，我们在回显用户信息时执行了一个延时的操作。假设已在web.xml配置文件中注册了该Servlet，现有两个用户a和b同时访问该Servlet（可以启动两个IE浏览器，或者在两台机器上同时访问），即同时在浏览器中输入：

a：http://localhost:8080/ServletTest/ConcurrentTest?Username=a

b：http://localhost:8080/ServletTest/ConcurrentTest?Username=b

如果用户b比用户a回车的时间稍慢一点，将得到如图2所示的输出：

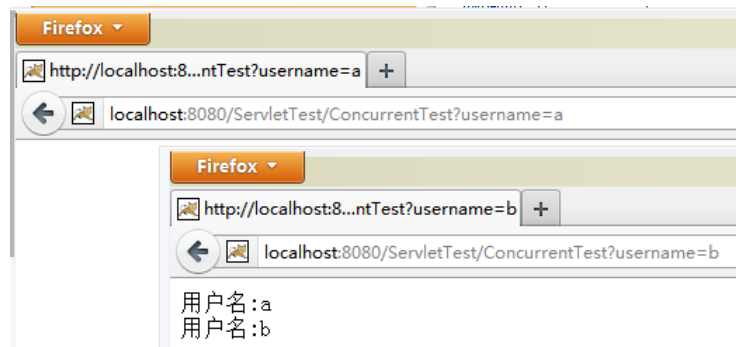


图2 a用户和b用户的浏览器输出

从图2中可以看到，Web服务器启动了两个线程分别处理来自用户a和用户b的请求，但是在用户a的浏览器上却得到一个空白的屏幕，用户a的信息显示在用户b的浏览器上。该Servlet存在线程不安全问题。下面我们就从分析该实例的内存模型入手，观察不同时刻实例变量output的值来分析使该Servlet线程不安全的原因。

Java的内存模型JMM (Java Memory Model) JMM主要是为了规定了线程和内存之间的一些关系。根据JMM的设计，系统存在一个主内存(Main Memory)，Java中所有实例变量都储存在主存中，对于所有线程都是共享的。每条线程都有自己的工作内存(Working Memory)，工作内存由缓存和堆栈两部分组成，缓存中保存的是主存中变量的拷贝，缓存可能并不总和主存同步，也就是缓存中变量的修改可能没有立刻写到主存中；堆栈中保存的是线程的局部变量，线程之间无法相互直接访问堆栈中的变量。根据JMM，我们可以将论文中所讨论的Servlet实例的内存模型抽象为图3所示的模型。

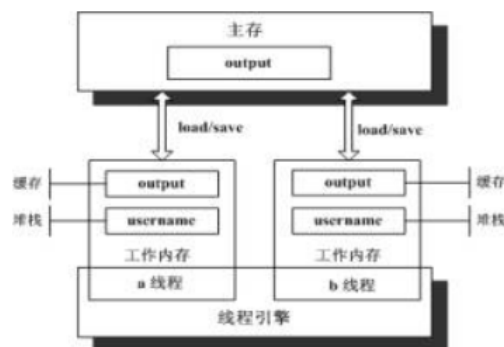


图3 Servlet实例的JMM模型

下面根据图3所示的内存模型，来分析当用户a和b的线程（简称为a线程、b线程）并发执行时，Servlet实例中所涉及变量的变化情况及线程的执行情况，如图4所示。

阅读排行榜

- 1. C++ 模板详解 (一) (227975)
- 2. Java运算符优先级(114447)
- 3. Java内存管理：深入Java内存区域(80644)
- 4. 【linux】提醒"libc.so.6: version `GLIBC_2.14' not found"系统的glibc版本太低(66395)
- 5. java.io.Serializable浅析(40751)

评论排行榜

- 1. 巨人网络的三道坑爹改错题(71)
- 2. C++ 模板详解 (一) (29)
- 3. C++ 模板详解 (二) (12)
- 4. java.lang.Object.clone()解读(10)
- 5. Java运算符优先级(7)

推荐排行榜

- 1. C++ 模板详解 (一) (44)
- 2. Java内存管理：深入Java内存区域(14)
- 3. Java运算符优先级(12)
- 4. Java Arrays.sort源代码解析(9)
- 5. Java跨平台原理(8)

调度时刻	a线程	b线程
T1	访问Servlet页面	
T2		访问Servlet页面
T3	output=a的输出username=a 休眠5000毫秒，让出CPU	
T4		output=b的输出（写回主存） username=b休眠5000毫秒，让出CPU
T5	在用户b的浏览器上输出a线程 的username的值,a线程终止。	
T6		在用户b的浏览器上输出b线程的 username的值,b线程终止。

图4 Servlet实例的线程调度情况

从图4中可以清楚的看到，由于b线程对实例变量output的修改覆盖了a线程对实例变量output的修改，从而导致了用户a的信息显示在了用户b的浏览器上。如果在a线程执行输出语句时，b线程对output的修改还没有刷新到主存，那么将不会出现图2所示的输出结果，因此这只是一种偶然现象，但这更增加了程序潜在的危险性。

设计线程安全的Servlet

通过上面的分析，我们知道了**实例变量不正确的使用是造成Servlet线程不安全的主要原因**。下面针对该问题给出了三种解决方案并对方案的选取给出了一些参考性的建议。

1、实现 SingleThreadModel 接口

该接口指定了系统如何处理对同一个Servlet的调用。如果一个Servlet被这个接口指定,那么在这个Servlet中的service方法将不会有二个线程被同时执行，当然也就不存在线程安全的问题。这种方法只要将前面的Concurrent Test类的类头定义更改为：

```
1 public class ConcurrentTest extends HttpServlet implements SingleThreadModel {
2     ...
3 }
```

javax.servlet.SingleThreadModel API及其翻译

Ensures that servlets handle only one request at a time. This interface has no methods.

确保servlet每次只处理一项请求。接口不含方法。

If a servlet implements this interface, you are *guaranteed* that no two threads will execute concurrently in the servlet's `service` method. The servlet container can make this guarantee by synchronizing access to a single instance of the servlet, or by maintaining a pool of servlet instances and dispatching each new request to a free servlet.

如果servlet实现了该接口，会确保不会有二个线程同时执行servlet的service方法。servlet容器通过同步化访问servlet的单实例来保证，也可以通过维持servlet的实例池，对于新的请求会分配给一个空闲的servlet。

Note that SingleThreadModel does not solve all thread safety issues. For example, session attributes and static variables can still be accessed by multiple requests on multiple threads at the same time, even when SingleThreadModel servlets are used. It is recommended that a developer take other means to resolve those issues instead of implementing this interface, such as avoiding the usage of an instance variable or synchronizing the block

of the code accessing those resources. This interface is deprecated in Servlet API version 2.4.

注意：SingleThreadModel不会解决所有的线程安全隐患。 例如，会话属性和静态变量仍然可以被多线程的多请求同时访问，即便使用了SingleThreadModel servlet。**建议开发人员应当采取其他手段来解决这些问题，而不是实现该接口，比如 避免实例变量的使用或者在访问资源时同步代码块。该接口在Servlet API 2.4中将不推荐使用。**

2、同步对共享数据的操作

使用synchronized 关键字能保证一次只有一个线程可以访问被保护的区段，在本论文中的Servlet可以通过同步块操作来保证线程的安全。同步后的代码如下：

```
1 public class ConcurrentTest extends HttpServlet {
2     PrintWriter output;
3     @Override
4     protected void service(HttpServletRequest request, HttpServletResponse response)
5         throws ServletException, IOException {
6         String username;
7         response.setContentType("text/html;charset=gb2312");
8         username=request.getParameter("username");
9         synchronized(this){
10             output=response.getWriter();
11             try {
12                 //为了突出并发问题，在这设置一个延时
13                 Thread.sleep(5000);
14                 output.println("用户名:"+username+"<BR>");
15             } catch (Exception e) {
16                 e.printStackTrace();
17             }
18         }
19     }
20 }
```

3、避免使用实例变量

本实例中的线程安全问题是实例变量造成的，只要在Servlet里面的任何方法里面都不使用实例变量，那么该Servlet就是线程安全的。

修正上面的Servlet代码，将实例变量改为局部变量实现同样的功能，代码如下：

```
1 public class ConcurrentTest extends HttpServlet {
2     @Override
3     protected void service(HttpServletRequest request, HttpServletResponse response)
4         throws ServletException, IOException {
5         PrintWriter output;
6         String username;
7         response.setContentType("text/html;charset=gb2312");
8         username=request.getParameter("username");
9         synchronized(this){
10             output=response.getWriter();
11             try {
12                 //为了突出并发问题，在这设置一个延时
13                 Thread.sleep(5000);
14                 output.println("用户名:"+username+"<BR>");
15             } catch (Exception e) {
16                 e.printStackTrace();
17             }
18         }
19     }
20 }
```



对上面的三种方法进行测试，可以表明用它们都能设计出线程安全的Servlet程序。但是，如果一个Servlet实现了SingleThreadModel接口，Servlet引擎将为每个新的请求创建一个单独的Servlet实例，这将引起大量的系统开销。SingleThreadModel在Servlet2.4中已不再提倡使用；同样如果在程序中使用同步来保护要使用的共享的数据，也会使系统的性能大大下降。这是因为被同步的代码块在同一时刻只能有一个线程执行它，使得其同时处理客户请求的吞吐量降低，而且很多客户处于阻塞状态。另外为保证主存内容和线程的工作内存中的数据的一致性，要频繁地刷新缓存,这也会大大地影响系统的性能。所以在实际的开发中也应避免或最小化 Servlet 中的同步代码；在Servlet中避免使用实例变量是保证Servlet线程安全的最佳选择。从Java 内存模型也可以知道，方法中的临时变量是在栈上分配空间，而且每个线程都有自己私有的栈空间，所以它们不会影响线程的安全。

小结

Servlet的线程安全问题只有在大量的并发访问时才会显现出来，并且很难发现，因此在编写Servlet程序时要特别注意。线程安全问题主要是由实例变量造成的,因此在Servlet中应避免使用实例变量。**如果应用程序设计无法避免使用实例变量，那么使用同步来保护要使用的实例变量，但为保证系统的最佳性能，应该同步可用性最小的代码路径。**


分类: [线程安全](#)

标签: [Servlet](#), [线程安全](#)

好文要顶

关注我

收藏该文



zero516cn

关注 - 4

粉丝 - 266

+加关注

6

0

« 上一篇: [Struts2请求处理流程及源码分析](#)
» 下一篇: [Servlet线程安全](#)

posted @ 2012-09-07 14:08 zero516cn 阅读(20622) 评论(4) 编辑 收藏

评论列表

- #1楼 2014-12-28 15:03 蓝枫居士

回复 引用

在
3、避免使用实例变量中,
已经去掉了实例变量 还需要
synchronized(this){} 这个保护么?

支持(0) 反对(0)
- #2楼 2014-12-28 15:33 蓝枫居士

回复 引用

惭愧呀,看了半天才看懂
output=response.getWriter();
有两个请求,自然就会有两个response
那么根据上面的代码 就会有两个output
如果并发执行且output为成员变量,那么就有可能两个线程的output都执行一个jsp文件 那两个数据自然就出现在一个页面喽

支持(0) 反对(0)
- #3楼 2014-12-28 15:44 蓝枫居士

回复 引用

如果我们只使用局部变量而不用实例变量,
就可以不用synchronized(this){} 这个保护

支持(0) 反对(0)
- #4楼 2018-10-25 03:30 wanxu

回复 引用

老哥，Tomcat中的ApplicationFilterChain貌似是每个Url Pattern的单例，想问下它是如何保证其成员变量（例如pos与n）的线程安全呢？这里我一直不理解

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

昵称：	Mi_Dad
-----	--------








[退出](#) [订阅评论](#)

【福利】学AI有奖：博客园&华为云 Modelarts 有奖训练营

- 深入研究Servlet线程安全问题[转贴]
- 深入研究Servlet线程安全问题
- 深入研究Servlet线程安全问题
- 深入研究Servlet线程安全问题
- Servlet 是否线程安全

- 首批非洲造智能手机问世了，售价不到200美元
- 900万吨粮食爆炸的灾难，元凶绝对让你意想不到
- 中国棉花种子在月球发芽：遗憾因过低气温并未存活太久
- 美国汽车协会实测：行人检测系统都是渣渣，包括特斯拉
- 美国贸易管制名单列入8家中国公司，盯视、海康威视、大华最新回应

» 更多新闻...