[**Servlet线程不安全是如何体现的？**](http://www.cnblogs.com/alaskan/p/4178533.html)

   在这个遍地框架的年代，我相信很多人对于底层的Servlet的深入了解肯定很少，但是大家肯定对于Servlet的线程安全肯定有所涉猎，也都能讲个大概。我也和大家一样，在一次与同事之间

的闲聊时，谈到了Servlet(PS:现在的工作中应用的框架就只有spring，其他俩个框架没有用，用的是Servlet。)突然说到线程安全的问题，我不假思索的说句，Servlet之所以线程不安全，是

因为Servlet共享了一个实例变量，所以在多线程的环境下容易产生线程不安全的问题。但是同事却抛出了另外一个问题：Servlet线程不安全的确是因为多线程环境下共享一个实例变量导致

的，在你设计一个Servlet的时候应该如何去避免线程安全问题呢？也许很多同学会顺口的答出用锁，锁住这个实例变量就能保证在多线程环境下的线程安全问题。但现在的要求是不能锁住该实

例变量，你能办到吗？这个就是知其然不知其所以然，所以变换一种要求就会显得无所适从。

先从Servlet的工作原理说起：首先简单解释一下Servlet接收和响应客户请求的过程，首先客户发送一个请求，Servlet是调用service()方法对请求进行响应的，通过源代码可见，service()方

法中对请求的方式进行了匹配，选择调用doGet,doPost等这些方法，然后再进入对应的方法中调用逻辑层的方法，实现对客户的响应。在Servlet接口和GenericServlet中是没doGet,doPost

等等这些方法的，HttpServlet中定义了这些方法，但是都是返回error信息，所以，我们每次定义一个Servlet的时候，都必须实现doGet或doPost等这些方法。

　　每一个自定义的Servlet都必须实现Servlet的接口，Servlet接口中定义了五个方法，其中比较重要的三个方法涉及到Servlet的生命周期，分别是上文提到的init(),service(),destroy()方

法。GenericServlet是一个通用的，不特定于任何协议的Servlet,它实现了Servlet接口。而HttpServlet继承于GenericServlet，因此HttpServlet也实现了Servlet接口。所以我们定Servle

t的时候只需要继承HttpServlet即可。

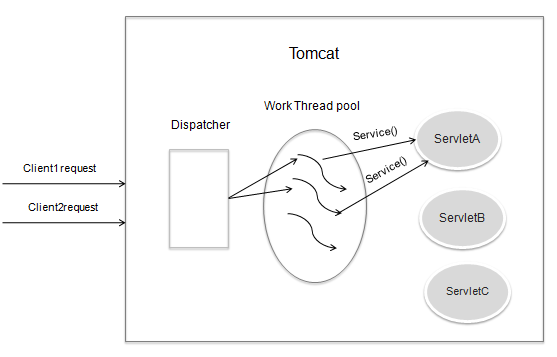
　　Servlet接口和GenericServlet是不特定于任何协议的，而HttpServlet是特定于HTTP协议的类，所以HttpServlet中实现了service()方法，并将请求ServletRequest,ServletResponse

强转为HttpRequest和HttpResponse。在Servlet整个生命周期中是由Tomcat来维护的，当客户端第一次发起请求的时候，会根据web.xml文件中的配置实例化一个Servlet,而在以后客户

端的每一次请求都会使用该实例来处理后续的工作，知道Tomcat停止该项目，这个Servlet才会被销毁，所占用的资源才会释放。

　　当客户端发来多个请求的时候，Servlet将采用多线程来解决这样的并发，而在Tomcat本身也维护了一个线程池来处理并发。线程池实际上是等待执行代码的一组线程叫做工作组线程

(Worker Thread)，Tomcat容器使用一个调度线程来管理工作组线程(Dispatcher Thead)。



我们还是来写个简单的Servlet实例来模拟下：

[复制代码](javascript:void(0);)

1 import java.io.IOException;

2 import java.io.PrintWriter;

3

4 import javax.servlet.http.HttpServlet;

5 import javax.servlet.http.HttpServletRequest;

6 import javax.servlet.http.HttpServletResponse;

7

8

9 /\*\*

10 \* @author zWX240091

11 \*

12 \*/

13 public class HelloWorldServlet extends HttpServlet

14 {

15 String message;

16 /\*\*

17 \*

18 \*/

19 private static final long serialVersionUID = 787553024399133588L;

20 public void service(HttpServletRequest request,HttpServletResponse response) throws IOException{

21 message =request.getParameter("message");

22 PrintWriter pw = response.getWriter();

23 try

24 {

25 Thread.sleep(5000);

26 }

27 catch (InterruptedException e)

28 {

29 e.printStackTrace();

30 }

31 pw.write("<div><strong>Hello World</strong>!</div>"+message);

32 pw.close();

33 }

34

35 }

[复制代码](javascript:void(0);)

在构建好的工程部署到本地Tomcat下，启动Tomcat，在打开俩个浏览器

分别访问1： http://localhost:8080/Servlet03/HelloWorld?message=helloA

分别访问2： http://localhost:8080/Servlet03/HelloWorld?message=helloB

然后分别刷新访问，然后你会惊喜的发现在访问第一个地址的页面打印出了helloB,在访问第二个地址的时候页面有时候会打印出helloA。这个就是高并发下的多线程的安全问题。

那如何不用加锁的情况下如何让高并发多线程的环境下线程安全呢？不知道大家注意到没有，在这个Servlet类里面，message这个属性是属于HelloWorldServlet这个实例变量，也就是说这

个实例变量是一个共享的实例变量，那么他所包含所有的属性都被这个实例变量共享了造成数据不匹配。我们是否可以将共享实例变量下的属性写到局部变量去呢？答案是可以的，将上面那段

代码中的属性message搬到Service中，在安全来的操作在访问下，你就会发现永远不会出现在传递过来的参数是helloA的时候页面会展示给你的是helloB，这样也就保证了能够线程安全了。

当然利用锁也是能够保证线程安全的，至于性能，就需要看你在什么样的环境去选择相应的方法去保证线程安全了。后面Google了下为什么局部变量为什么不会被共享，解释如下：多线程下每

个线程对局部变量都会有自己的一份copy，这样对局部变量的修改只会影响到自己的copy而不会对别的线程产生影响，线程安全的。但是对于实例变量来说，由于servlet在Tomcat中是以单

例模式存在的，所有的线程共享实例变量。多个线程对共享资源的访问就造成了线程不安全问题。

资料参考地址：[深入理解Servlet线程安全问题](http://blog.csdn.net/lcore/article/details/8974590)http://blog.csdn.net/lcore/article/details/8974590

[Servlet生命周期与工作原理](http://www.cnblogs.com/cuiliang/archive/2011/10/21/2220671.html)http://www.cnblogs.com/cuiliang/archive/2011/10/21/2220671.html