JSP 异常处理 JSP 国际化

JSP 调试

要测试/调试一个JSP或servlet程序总是那么的难。JSP和Servlets程序趋向于牵涉到大量客户端/服务器之间的交互，这很有可能会产生错误，并且很难重现出错的环境。

接下来将会给出一些小技巧和小建议，来帮助您调试程序。

使用System.out.println()

System.out.println()可以很方便地标记一段代码是否被执行。当然，我们也可以打印出各种各样的值。此外：

- 自从System对象成为Java核心对象后，它便可以使用在任何地方而不用引入额外的类。使用范围包括Servlets，JSP，RMI，EJB's，Beans，类和独立应用。
- 与在断点处停止运行相比，用System.out进行输出不会对应用程序的运行流程造成重大的影响，这个特点在定时机制非常重要的应用程序中就显得非常有用了。

接下来给出了使用System.out.println()的语法：

```
System.out.println("Debugging message");
```

这是一个使用System.out.print()的简单例子：

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head><title>System.out.println</title></head>
<body>
<c:forEach var="counter" begin="1" end="10" step="1" >
   <c:out value="${counter-5}"/></br>
   <% System.out.println( "counter= " +
                     pageContext.findAttribute("counter") ); %>
</c:forEach>
</body>
</html>
```

现在，如果运行上面的例子的话，它将会产生如下的结果：

```
-4
-3
-2
-1
0
1
2
3
```

```
4
5
```

如果使用的是Tomcat服务器，您就能够在logs目录下的stdout.log文件中发现多出了如下内容：

```
counter=1
counter=2
counter=3
counter=4
counter=5
counter=6
counter=7
counter=8
counter=9
counter=10
```

使用这种方法可以将变量和其它的信息输出至系统日志中，用来分析并找出造成问题的深层次原因。

使用JDB Logger

J2SE日志框架可为任何运行在JVM中的类提供日志记录服务。因此我们可以利用这个框架来记录任何信息。

让我们来重写以上代码，使用JDK中的 logger API：

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@page import="java.util.logging.Logger" %>

<html>
<head><title>Logger.info</title></head>
<body>
<% Logger logger=Logger.getLogger(this.getClass().getName());%>

<c:forEach var="counter" begin="1" end="10" step="1" >
   <c:set var="myCount" value="${counter-5}" />
   <c:out value="${myCount}"/></br>
   <% String message = "counter="
                  + pageContext.findAttribute("counter")
                  + " myCount="
                  + pageContext.findAttribute("myCount");
                  logger.info( message );
   %>
</c:forEach>
</body>
</html>
```

它的运行结果与先前的类似，但是，它可以获得额外的信息输出至stdout.log文件中。在这我们使用了logger中的info方法。下面我们给出stdout.log文件中的一个快照：

```
24-Sep-2013 23:31:31 org.apache.jsp.main_jsp _jspService
INFO: counter=1 myCount=-4
24-Sep-2013 23:31:31 org.apache.jsp.main_jsp _jspService
INFO: counter=2 myCount=-3
24-Sep-2013 23:31:31 org.apache.jsp.main_jsp _jspService
INFO: counter=3 myCount=-2
24-Sep-2013 23:31:31 org.apache.jsp.main_jsp _jspService
INFO: counter=4 myCount=-1
24-Sep-2013 23:31:31 org.apache.jsp.main_jsp _jspService
INFO: counter=5 myCount=0
24-Sep-2013 23:31:31 org.apache.jsp.main_jsp _jspService
INFO: counter=6 myCount=1
24-Sep-2013 23:31:31 org.apache.jsp.main_jsp _jspService
INFO: counter=7 myCount=2
24-Sep-2013 23:31:31 org.apache.jsp.main_jsp _jspService
INFO: counter=8 myCount=3
24-Sep-2013 23:31:31 org.apache.jsp.main_jsp _jspService
INFO: counter=9 myCount=4
24-Sep-2013 23:31:31 org.apache.jsp.main_jsp _jspService
INFO: counter=10 myCount=5
```

消息可以使用各种优先级发送，通过使用sever()，warning()，info()，config()，fine()，finer()，finest()方法。finest()方法用来记录最好的信息，而sever()方法用来记录最严重的信息。

使用Log4J 框架来将消息记录在不同的文件中，这些消息基于严重程度和重要性来进行分类。

调试工具

NetBeans是树形结构，是开源的Java综合开发环境，支持开发独立的Java应用程序和网络应用程序，同时也支持JSP调试。

NetBeans支持如下几个基本的调试功能：

- 断点
- 单步跟踪
- 观察点

详细的信息可以查看NetBeans使用手册。

使用JDB Debugger

可以在JSP和servlets中使用jdb命令来进行调试，就像调试普通的应用程序一样。

通常，我们直接调试sun.servlet.http.HttpServer 对象来查看HttpServer在响应HTTP请求时执行JSP/Servlets的情况。这与调试applets非常相似。不同之处在于，applets程序实际调试的是sun.applet.AppletViewer。

大部分调试器在调试applets时都能够自动忽略掉一些细节，因为它知道如何调试applets。如果想要将调试对象转移到JSP身上，就需要做好以下两点：

- 设置调试器的classpath，让它能够找到sun.servlet.http.Http-Server 和相关的类。

- 设置调试器的classpath，让它能够找到您的JSP文件和相关的类。

设置好classpath后，开始调试sun.servlet.http.Http-Server 。您可以在JSP文件的任意地方设置断点，只要你喜欢，然后使用浏览器发送一个请求给服务器就应该可以看见程序停在了断点处。

使用注释

程序中的注释在很多方面都对程序的调试起到一定的帮助作用。注释可以用在调试程序的很多方面中。

JSP使用Java注释。如果一个BUG消失了，就请仔细查看您刚注释过的代码，通常都能找出原因。

客户端和服务器的头模块

有时候，当JSP没有按照预定的方式运行时，查看未加工的HTTP请求和响应也是很有用的。如果对HTTP的结构很熟悉的话，您可以直接观察request和response然后看看这些头模块到底怎么了。

重要调试技巧

这里我们再透露两个调试JSP的小技巧：

- 使用浏览器显示原始的页面内容，用来区分是否是格式问题。这个选项通常在View菜单下。
- 确保浏览器在强制重新载入页面时没有捕获先前的request输出。若使用的是Netscape Navigator浏览器，则用Shift-Reload；若使用的是IE浏览器，则用Shift-Refresh。

← JSP 异常处理 JSP 国际化 →