12.9 The isThreadSafe Attribute

The isThreadSafe attribute controls whether the servlet that results from the JSP page will allow concurrent access (the default) or will guarantee that no servlet instance processes more than one request at a time (isThreadSafe="false"). Use of the isThreadSafe attribute takes one of the following two forms.

```
<%@ page isThreadSafe="true" %> <%-- Default --%>
<%@ page isThreadSafe="false" %>
```

Unfortunately, the standard mechanism for preventing concurrent access is to implement the SingleThreadModel interface (Section 3.7). Although SingleThreadModel and isThreadSafe="false" were recommended in the early days, recent experience has shown that SingleThreadModel was so poorly designed that it is basically useless. So, you should avoid isThreadSafe and use explicit synchronization instead.

Core Warning

Do not use isThreadSafe. Use explicit synchronization instead.

To understand why isThreadSafe="false" is a bad idea, consider the following non-thread-safe snippet to compute user IDs. It is not thread safe since a thread could be preempted after reading idNum but before updating it, yielding two users with the same user ID.

```
<%! private int idNum = 0; %>
<%
String userID = "userID" + idNum;
out.println("Your ID is " + userID + ".");
idNum = idNum + 1;
%>
```

The code should have used a synchronized block. This construct is written

```
synchronized(someObject) { ... }
```

and means that once a thread enters the block of code, no other thread can enter the same block (or any other block marked with the same object reference) until the first thread exits. So, the previous snippet should have been written in the following manner.

```
<%! private int idNum = 0; %>
<%
synchronized(this) {
  String userID = "userID" + idNum;
  out.println("Your ID is " + userID + ".");
  idNum = idNum + 1;
}
%>
```

There are two reasons why this explicitly synchronized version is superior to the original version with the addition of <%@ page isThreadSafe="false" %>.

First, the explicitly synchronized version will probably have much better performance if the page is accessed frequently. The reason is that most JSP pages are not CPU limited but are I/O limited. So, while the system waits for I/O (e.g., a response from a database, the result of an EJB call, output sent over the network to the user), it should be doing something else. Since most servers implement `SingleThreadModel` by queueing up requests and handling them one at a time, high-traffic JSP pages can be *much* slower with this approach.

Even worse, the version that uses `SingleThreadModel` might not even get the right answer! Rather than queueing up requests, servers are permitted to implement `SingleThreadModel` by making a pool of servlet instances, as long as no instance is invoked concurrently. This, of course, totally defeats the purpose of using fields for persistence, since each instance would have a different field (instance variable), and multiple users could still get the same user ID. Defining the `idNum` field as `static` does not solve the problem either; the `this` reference would be different for each servlet instance, so the protection would be ineffective.

These problems are basically intractable. Give up. Forget `SingleThreadModel` and `isThreadSafe="false"`. Synchronize your code explicitly instead.

[Team LiB] ◂ PREVIOUS NEXT ▸