**SUMMARY**

A concurrency bug in Eclipse 3.6 Platform Debug.

**DETAILS**

Some details can also be found at: <https://bugs.eclipse.org/bugs/show_bug.cgi?id=298648>

This bug is due to a data race.

|  |  |
| --- | --- |
| Thread1 (DebugModelContextBindingManager.java) | Thread2 (DebugModelContextBindingManager.java) |
| if (modelIdentifier != null && contextId != null) {  List contextIds = (List) fModelToContextIds.get(modelIdentifier);  if (contextIds == null) {  contextIds = new ArrayList();  fModelToContextIds.put(modelIdentifier, contextIds);  }  contextIds.add(contextId); | if (modelIdentifier != null && contextId != null) {  List contextIds = (List) fModelToContextIds.get(modelIdentifier);  if (contextIds == null) {  contextIds = new ArrayList();  fModelToContextIds.put(modelIdentifier, contextIds);  }  contextIds.add(contextId); |
| 0-lock | 0-lock |

1) The DebugModelContextBindingManager is prone to race conditions. There a couple of unprotected maps which are read and modified in response to debug events. Since debug event listeners can be called on any thread, this is a source of race conditions which I observed with a DSF-based debugger.

2) The activating of contexts and views is triggered off of a change in active debug context. Deactivating of contexts is triggered upon a launch termination. However, if a context is selected after a launch is terminated, there is no guard to prevent the context from being re-activated.

This problem is masked by the fact that for the standard debug model, only the stack frame is used to activate contexts (which I believe is also a bug).

3) There is a different race condition in ViewContextManager where the context-bound views are also activated by a debug context activation. However, activating views is only enabled upon a suspend event as received by an ISuspendTriggerListener. So if the model calls ISuspendTriggerListener after the debug context activate event, the views don't get activated properly.