Developing for Sakai in Eclipse: How to debug Sakai (Tomcat) using Eclipse

Debugging Sakai with Eclipse is really easy.
Tell Tomcat to listen for Debugger connections
In ‹tomcat›/bin/startup.sh -- that's the script that launches Sakai -- add two environment variables and add an argument to the actual launch command:

```
export JPDA_ADDRESS=8000
export JPDA_TRANSPORT=dt_socket
#add "jpda" below:
exec "$DIR/$EXEC" jpda start "$@"
```

That's how you tell your Tomcat to listen for debugger traffic. On your next startup.sh, you should see "Listening for transport dt_socket at address:8000" in your catalina.out. (Of course you can use whatever port is most appropriate for your situation, 8000 is not set in stone.) Now, configure your Eclipse to do the debugging:
Tell Eclipse how to connect to Tomcat
It should be obvious that the source code you're editing within eclipse, well, that's the code that Tomcat should be running. Otherwise everybody will likely get very confused! It's how you can expect Eclipse to keep track of the line numbers in the source files, to show you each line as it executes, in context.

With Eclipse launched (this can be on a completely different machine from where Sakai is actually running) set up the connection:

- Run menu -› open debug dialog -› double-click "remote java app"

- Specify which host and port to connect to (localhost and 8000 are the defaults, but you can hook up to a remote machine on whichever port you used above in JPDA_ADDRESS)
- Source tab -› add all java projects that you want to step through

Connect Eclipse to Tomcat
Now that Eclipse is ready to connect, and Tomcat is listening for connections, go ahead and connect:

- Set some break-points in your code! In Eclipse, right-click along the left margin of any java code window (the little dot that appears tells you you've got that line set as a breakpoint)

- Connect to your already-running Sakai instance of Tomcat by using the Run menu -› Debug
- As your Sakai percolates along, when it gets to the breakpoint (you can set many, whichever one it trips first will stop Tomcat in its tracks) Eclipse will ask if you want to switch to the DEBUG perspective.

"This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporated views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?"

So don't forget: when you've set breakpoints, and you've opened a debug connection, Tomcat will STOP and your browser will appear to hang... that's because Eclipse is waiting for you to tell it whether to step into the next line of code, to dive into your variables, and so forth!

Now, debug galore!
There, you can step-into, step-over, step-out... and check or modify variables... and review your call stack... and lots more!

Unique solution ID: #1060
Author: will trillich
Last update: 2008-06-21 14:39