Debugging Kelp projects

Table of Contents

Chapter 1. Introduction

One of the advantages of using an IDE for developing applications and components for Enhydra, is that the IDEs provide tools for debugging.

Chapter 2. Debugging with NetBeans

NetBeans supports remote debugging using the Java Platform Debugger Architecture (JPDA). The JPDA enables you to set breakpoints on classes, threads, and variables; to set conditional breakpoints; and to examine the value of an expression. The JPDA is the default debugger if you are running the Java 2 Platform, Standard Edition (J2SE), SDK 1.3.

The Debugging workspace automatically loads when you start a debugging session. By default, this workspace includes three windows:

- The Debugger window with separate tabs for managing breakpoints, variables, watches, and threads
- The Output window for displaying messages from the debugger
- The Source Editor for showing the line in the source code where the program is stopped

To debug using SharedMemoryAttach:

The SharedMemoryAttach method may be faster

- To debug an application using SharedMemoryAttach, launch an instance of Enhydra locally with the following command:

  ```
  multiserver -debug -jdwp-args "transport=dt_shmem,server=y,suspend=n,
  address=localdebug"
  ```

- In NetBeans, attach to the VM by choosing Debug|Attach to VM from the menus.
- In the Attach to VM dialog, set the following options to start debugging:
  - Debugger Type: Default debugger (JPDA)
  - Connector: SharedMemoryAttach
  - Transport: dt_shmem
  - Name: localdebug (this is specified in the command argument used to launch the Enhydra server)
- Deploy and start the application to debug.
- Open the application code in NetBeans to add watches or breakpoints.
- Access the application in a browser to exercise the code you are debugging.

To debug using SocketAttach:

By default, when you start the Enhydra server with the -debug argument, the server is configured for debugging via a socket attachment. This may be slower.

- To debug an application using SharedMemoryAttach, launch an instance of Enhydra locally with the following command:

```
multiserver -debug
```

- In NetBeans, attach to the VM by choosing Debug|Attach to VM from the menus.
- In the Attach to VM dialog, set the following options to start debugging:
  - Debugger Type: Default debugger (JPDA)
  - Connector: SocketAttach
  - Transport: dt_socket
  - Host: Enter the name of the host upon which the Enhydra server was started (e.g., localhost)
  - Port: Enter the port assigned for debugging
- Deploy and start the application to debug.
- Open the application code in NetBeans to add watches or breakpoints.
- Access the application in a browser to exercise the code you are debugging.

Chapter 3. Debugging with JBuilder

The only requirement for debugging applications from JBuilder is that you must start the Enhydra server from your JBuilder project. Before you debug an application, make sure you can run the Kelp sample project successfully. To do this, you will need to run the XML Compiler and the Kelp Deployer.

To debug an Enhydra application:

- Make sure the Enhydra server is not already running.
- Open the Kelp sample project.
- Select TableServlet.java in the kelp.webapp.presentation package.
- Locate the for loop of the clearTable() method and click the left edge of the editor to put a breakpoint on the line that reads table.appendChild(newRow);.

  After you set the breakpoint, the line appears in red.
- From the menu, choose Run|Debug.

  This starts the debugger. The debugger opens a command window even if you have selected Execution Log for console I/O.
- Open the table page in your browser by opening http://localhost:9000, where 9000 is the specified port and clicking Table Page.

  This triggers the breakpoint, as shown in Figure 9.

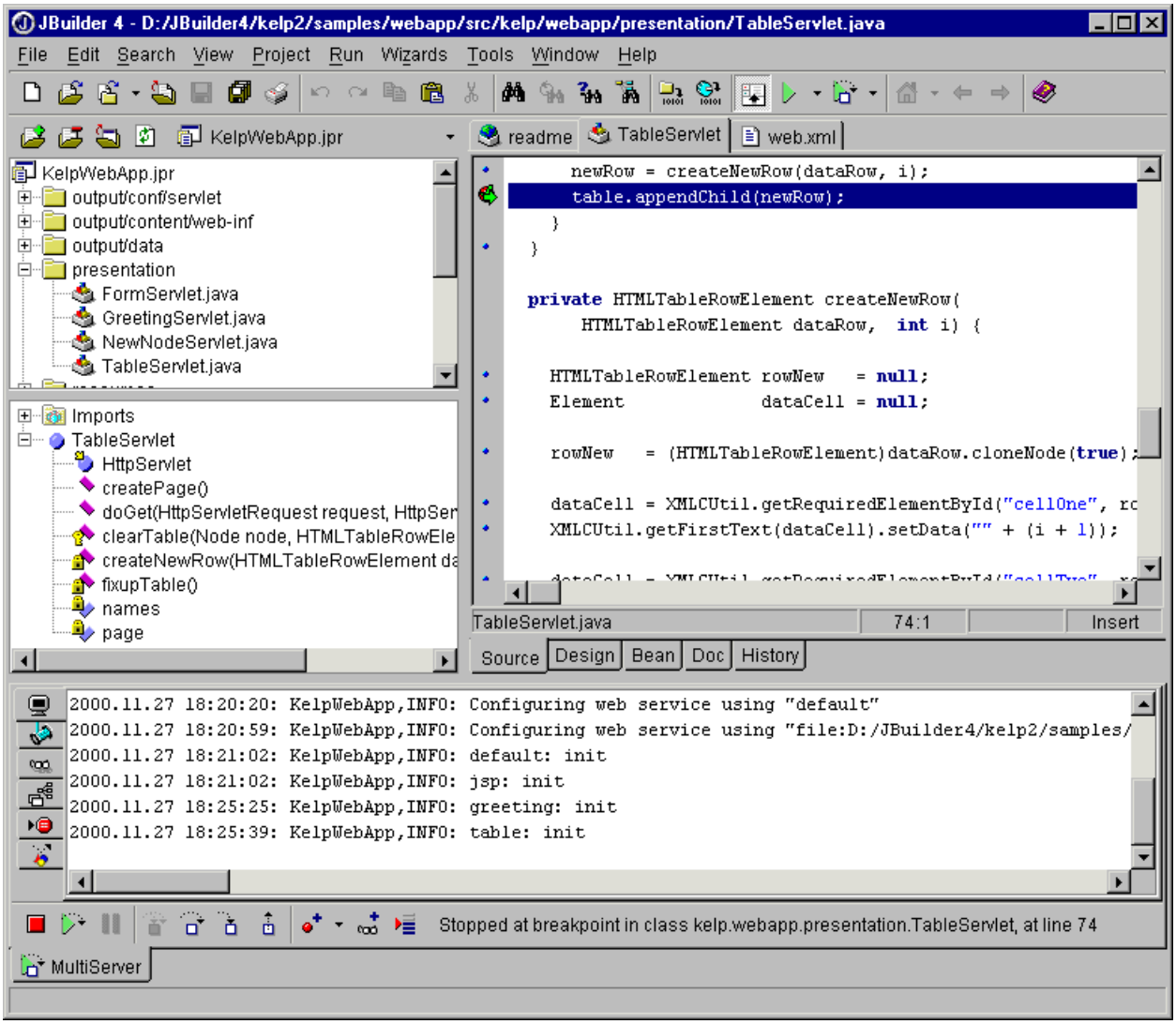

Figure 9: Debugging an application in JBuilder

You can set the value of i to 4 to change the number of rows generated in the table. Your browser might time-out if you keep the program suspended for too long.