JakartaTurbineIDEDebugging

Debugging Turbine using your IDE's debugger

There are two ways that you can choose to debug you app using your IDE. First, you can run your servlet container in your IDE. We will call this local debugging. The second option is to start your serlvet container in a seperate process and attach to it with your IDE using JDPA. We will call this option remote debugging. Of the two, remote debugging is often the easiest to configure.

Configuring your servlet container for remote debugging

For remote debugging the serlvet container will be configured the same way no matter what IDE you are using.

Remote debugging with Tomcat

Two environment variables need to be set before catalina.[sh|bat] is called.

- JPDA_TRANSPORT - Possible values: dt_shmem and dt_socket
- JPDA_ADDRESS

For a solution that works on any platform, I would suggest that you use JPDA_TRANSPORT=dt_socket. JPDA_ADDRESS is the socket number that will accept debugging connections. I use JPDA_ADDRESS=8000.

Next, you will need to start tomcat a little differently. Instead of using 'catalina run' or 'catalina start' you will need to use 'catalina jpda run' or 'catalina jpda start'.

Remote debugging with JBoss

There are probably a number of ways to do this, but the following works for JBoss (and similar changes should work for most other Java-based servlet containers, such as iPlanet, etc.). These instructions assume you can read a .bat file on Win32 or a shell script in *nix.

Make a copy of the run.bat (or run.sh) file in the <jboss>/bin directory. Call it debug.bat (or debug.sh). Find the section of the file where JAVA_OPTS is set. Add options to invoke Java in debug mode (you can, BTW, do this with any Java application). In the debug.bat, for example, change the line to this:

```
 set JAVA_OPTS=-classic -Xdebug -Xnoagent -Djava.compiler=NONE -Xrunjdwp:transport=dt_socket,address=8787,
server=y,suspend=y %JAVA_OPTS%
```

Note that the

```
 suspend=y
```

means that, on startup, the process will wait for you to connect to it with a debugger. You can change this by specifying

```
 suspend=n
```

instead.

Configuring you IDE for remote debugging

Remote debugging with IntelliJ

#Open the run/debug configurations window.
#Click on the Remote tab
#click the + to add a new configuration
#Set the name to something like "Tomcat - Socket"
#Debugger mode = attach
#transport = socket
#host = <hostname of the machine running tomcat that you want to attach to>
#port = <whatever you set for the listening socket on your serlvet container>

- If you want to be able to step through the source code of Turbine during your debugging, you will need to include the source for Turbine in the sourcepath tab of your project. The same applies for Torque.

Remote debugging with Eclipse

#Open a Debug perspective.
#Choose

```
Debug...
```

from the

```
Run
```

menu.
#In the dialog that opens, click

```
Remote Java Application
```

in the tree control.
#Click

```
New
```

.
#Name = <whatever you want to call it>
#Project = <select your project>
#Connection Type = Standard (Socket Attach)
#host = <hostname of the machine running tomcat that you want to attach to>
#port = <whatever you set for the listening socket on your serlvet container>

In version 2.0.2 of Eclipse, getting the debugger to let you connect to the Turbine and Torque source code is fairly tricky. (Later versions may make this easier.) The easiest way is to build Eclipse projects for them and compile the code directly. You are on your own on that one.

Another option is to link to the code during debugging. When you come to a class that Eclipse doesn't have the source for, it will give you a button to attach source to it. This, unfortunately, is not as easy as it sounds, primarily because the source to be attached *must* be contained in a .jar or .zip file. So, do this:

#Create a jar file containing the <turbine>/src/java and <torque>/src/java directories. In this example, the resulting jar is

```
<turbine>/target/turbine-src.jar
```

.
#When you click on the

```
Attach Code...
```

button, you'll get a confusing dialog.
#In the top field, you need to define a variable. You can do this a lot of ways. One way is to define

```
TURBINE_HOME
```

to be the location of your turbine root. Then add an "extension" of

```
target/turbine-src.jar
```

.
#Assuming you created your .jar file such that the internal heirarchy contains the

```
org
```

directory, you can leave the second field blank.

Configuring your servlet container for local debugging

As mentioned above, you can also use your IDE to launch your servlet container in the debugger. This is easier to do with certain combinations of IDE and servlet container than others. In most cases, you can look in the launch script for the container to figure out what the proper settings need to be.

Local debugging with Eclipse and JBoss

#Open a Debug perspective.
#Choose

```
Debug...
```

from the

```
Run
```

menu.
#In the dialog that opens, click

```
Java Application
```

in the tree control.
#Click

```
New
```

.
#Keep the defaults with the following exceptions:
#Main tab

#On the Classpath tab, add <jboss>/bin/run.jar
#Set other tabs to taste.

Local debugging with Eclipse and JBoss

To debug JBOSS/TOMCAT app in Eclipse use the plugin available at the following site:

http://www.genuitec.com/products_easie.htm

Local debugging with Eclipse and TomCat

To debug TOMCAT webapp in Eclipse use the plugin available at the following site:

http://www.sysdeo.com/eclipse/tomcatPlugin.html

CategoryJakartaTurbine2HowTo