Debugging with WebSphere Studio Device Developer

Technical Information Paper – v1.0

8 November 2004

Authors:

Name	Contact eMail
Sue Estrada	wctmepvc@us.ibm.com

Overview of debugging with WebSphere Studio Device Developer

Debugging is an essential step in the process of software development. A program debugger provides you with a tool to assist you in performing these tasks. It helps to eliminate the need to find errors through visual inspection of the code or using other techniques. The debugger in WebSphere® Studio Device Developer (WSDD) enables you to control the execution of your program by setting breakpoints, suspending launched programs, stepping through code, and examining the contents of variables. These functions enable you to detect and diagnose errors in your programs.

In addition, WSDD provides a set of tools that enables you to debug a program independent of its physical location. You can debug a program locally while it is running on the same system as WSDD in a Java™ Virtual Machine (JVM) or in an emulator that is integrated into WSDD. You can also debug the program remotely while it is running on a physical device accessible through a network or while it is running in an emulator that is not integrated into WSDD, such as a UEI emulator.

Local Debugging

Local debugging is the method most often used by developers. Before you can debug however, you must create your Java program, build it and then create a launch configuration for it. Launch configurations are used in WSDD to run programs. You can run your program as a Java application or you can choose to simulate execution on your device by using a **Java on Device** launch configuration. In the case of a **Java on Device** launch, there are two types of launch configurations that are used to associate a project with a particular build and a particular device. A **MIDlet Suite** launch configuration is used to launch a MIDlet. A **Java on Device** launch configuration is used to run a non-MIDlet application.

For local debugging, the same launch configuration can be used to run or to debug a program. To debug, launch the program using the **Run > Debug** menu item on the Workbench menu bar. Launching the program in this way establishes a connection between the debugger and the Java program that you are launching. You can then use breakpoints, stepping, or expression evaluation to debug your program.

Remote Debugging

Remote debugging is made possible through the client/server design of the debugger. This design makes it possible to launch a Java program on a device or a computer on the network and then debug it from a workstation running WSDD. This is particularly useful when developing applications for small devices that are not capable of running a development platform or a debugger themselves. In this type of scenario, a Debug Info Server, j9dbgserv, is available for use as a repository for debug information and symbol files. When j9dbgserv is used, the VM must be configured to connect to this Debug Info Server using a particular TCP/IP port. Debug information and symbol files can be

resident in the same location where the remote program is run; however this is typically not practical when dealing with resource-constrained devices.

When debugging remotely, there are two types of launch configurations that you can use: **Remote Java Application** and **Remote Java on Device**. Use the **Remote Java Application** launch configuration when you are debugging an application that is running on a remote VM. Because the application is started on the remote system, the launch configuration does not specify the usual information about the JRE, program arguments, or VM arguments. Instead, information about connecting to the application is supplied. To debug a program remotely in this manner you must be able to launch the program in debug mode so that it waits for a connection from your debugger.

The **Remote Java on Device** launch configuration starts the Java debugger that is configured to wait for a connection from the device. The following figure shows the architecture of the **Remote Java Application** and **Remote Java on Device** launches.

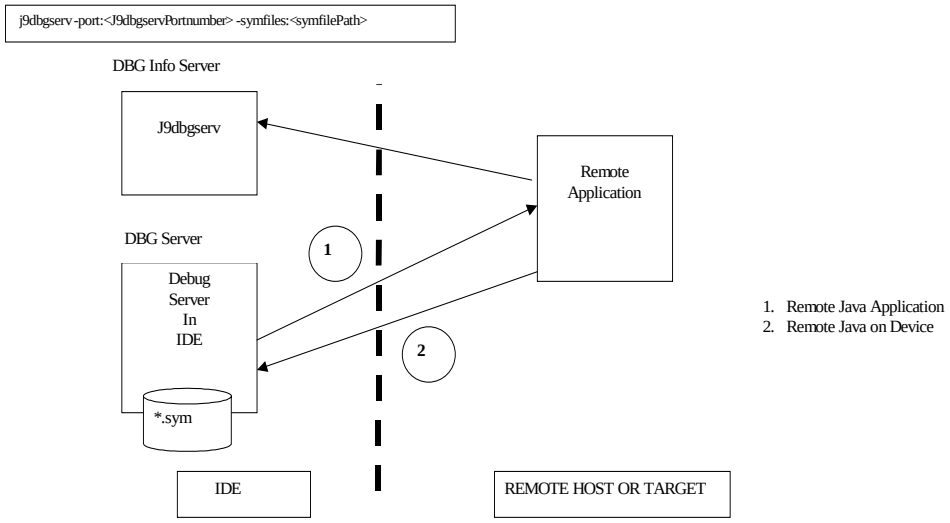

Figure 1: Launch Architecture

Remote Java on Device Example

To debug a program remotely using the **Remote Java on Device**, perform the following steps:

1. Manually start a j9dbgserv Debug Info Server in a command window using the following command:

   j9dbgserv -port*:*<*J9dbgservPortnumber*> -symfiles*:*<*symfilePath*>

   where the *symfilePath* is required to debug JXE-build applications and the *J9dbgservPortnumber* defaults to 8888.

2. Create and configure a **Remote Java on Device** debug launch configuration on WSDD. Click **Run>Debug** from the Workbench menu bar, select **Remote Java on Device** for the configuration and then click **New**. Enter a value for WSDD **Host** IP address (or localhost) & WSDD Host debug **Port** in the debug connection properties port section of the dialog.

3. Manually deploy the application to the remote device and start the application from a command window, adding the following debug options for use by the VM:

   -debug:address =<***WSDD Host IP*>:<*WSDD Host port***> -Xrdbginfo:<***WSDD HOST IP*>:<*J9dbgservPortnumber***>**

Note: For some devices, such as for PocketPC, WSDD has integrated the device in such way that the user can debug using a **Java on Device** launch configuration. Under such circumstances, WSDD performs the steps required in the **Remote Java on Device** example given above in a manner that is transparent to the user. This enables programs for the PocketPC to be debugged using the local scenario.

The Debug perspective in WSDD contains several views, including one for Debug, Variables/Breakpoints/Expressions/Display, Editor, Outline, and Console/Tasks. When debugging a PocketPC program, additional views appear in the Debug perspective for the device, as shown in the following figure.

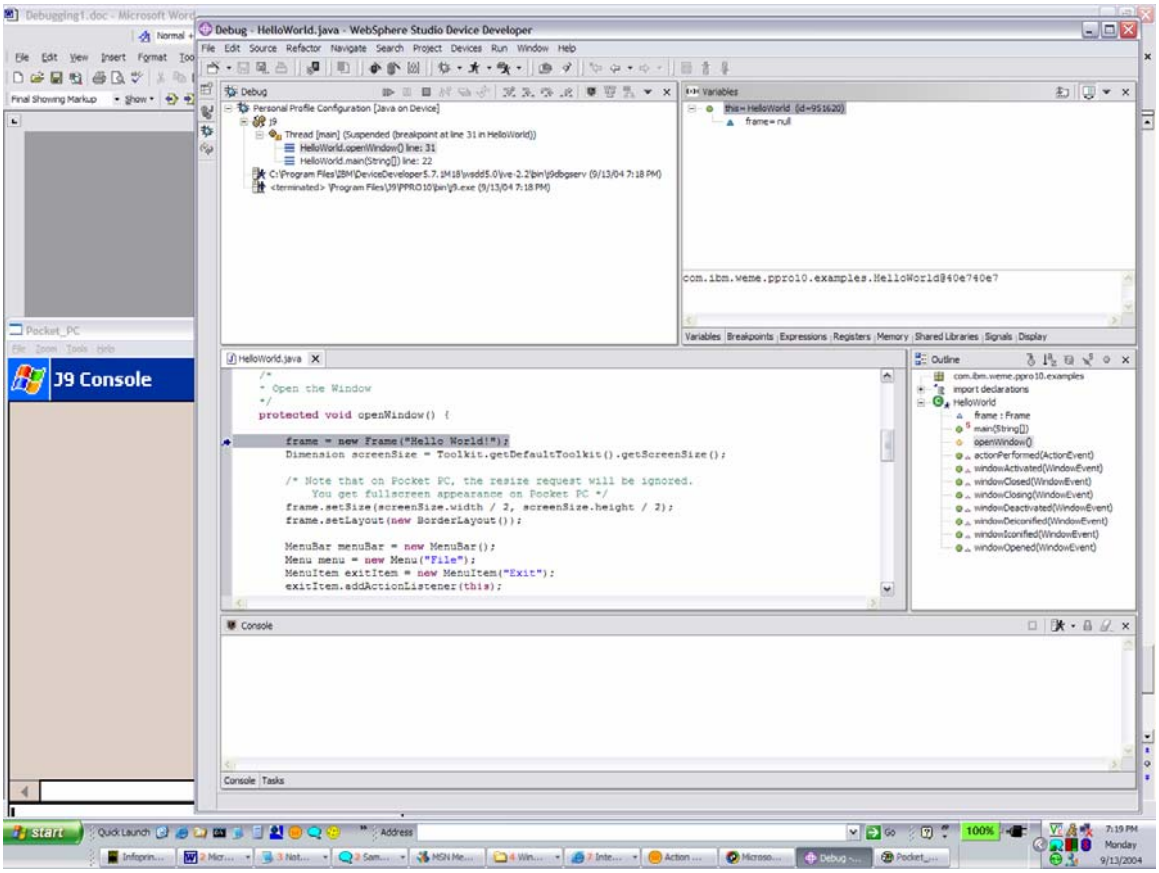

Figure 2: WSDD Debug Perspective

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

WebSphere

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product and service names may be trademarks or service marks of others.