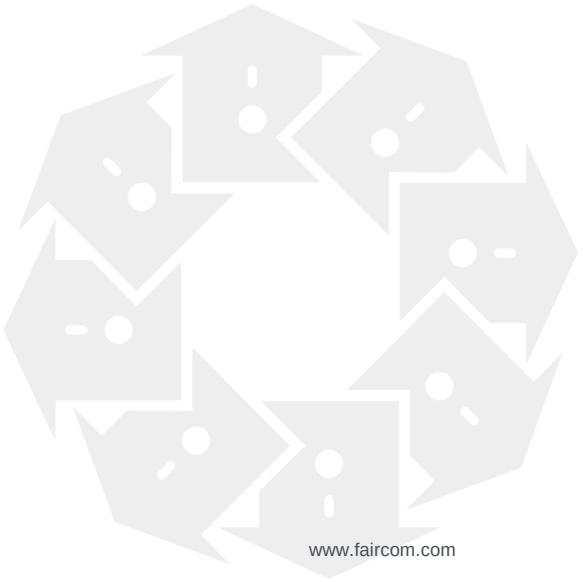
FairCom White Paper

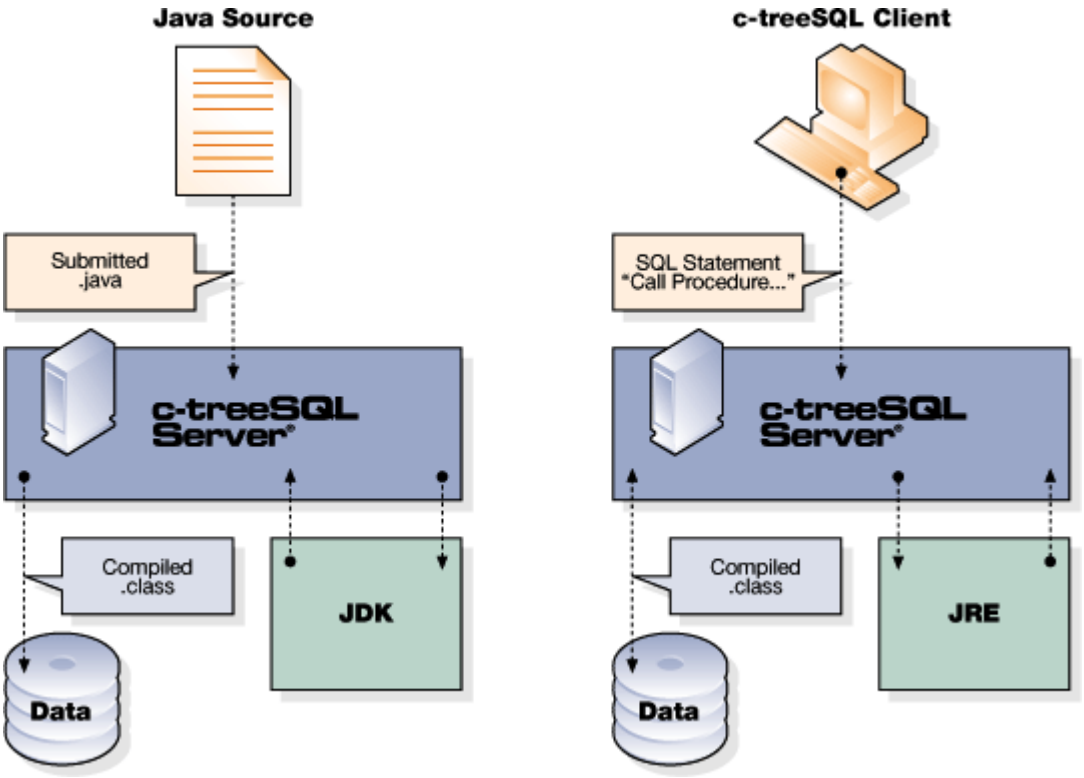
Stored Procedure Debugging in Java

Contents



1. Java Stored Procedures

c-treeACE SQL includes support for Java stored procedures, triggers, and user-defined functions. Stored procedures, triggers, and user-defined functions provide the ability to write Java routines that contain SQL statements and store those routines within a database.



Java Stored Procedures

A stored procedure is a snippet of Java code embedded in an SQL `CREATE PROCEDURE` statement. The Java snippet can use all standard Java features as well as use Java classes provided with c-treeACE SQL for processing any number of SQL statements.

Stored procedures are compiled and stored in bytecode format for direct execution without recompilation and parsing. Java stored procedures can return result sets to calling applications or procedures.

Java Triggers

A trigger is a special type of stored procedure that helps ensure referential integrity for a database. Like stored procedures, triggers also contain Java code (embedded in a CREATE TRIGGER statement) and use c-treeACE Java classes. However, triggers are automatically invoked ("fired") by certain SQL operations (an insert, update, or delete operation) on the trigger's target table.

Java User Defined Functions

Scalar functions are an integral part of the support provided by c-treeACE SQL for query expression. c-treeACE SQL provides many built-in scalar functions that transform data in different ways. Sometimes, however, there is a need for a custom-transformation of the data—a transformation that is not available with any of the provided functions. This is solved by user-defined scalar functions (UDFs)—a scalar function that is defined by the user. c-treeACE SQL UDFs allow the user to define their own functions to transform data in some custom manner.

The user defines functions by creating Java Stored Functions, modules written in Java that are similar to the ones written for stored procedures and triggers. The java code snippet contained in the User Defined Scalar Function definition is processed by c-treeACE SQL into a Java class definition and stored in the database in text and compiled form.

Published 11/16/2018

2. Debugging in Java

c-treeACE SQL allows Java debugging tools, such as JSwat, to connect and directly debug stored procedure routines. JSwat is an open-source GUI Java debugger, which can be downloaded from https://github.com/nlfiedler/jswat (https://github.com/nlfiedler/jswat).

This section includes information about configuring your environment to enable debugging with JSwat (page 3) and provides an example of how to use JSwat (page 4).

2.1 Enabling Java Debugging Tools

c-treeACE SQL allows Java debugging tools to connect and directly debug stored procedure routines. To enable this feature, follow the following steps.

1. Add the following keywords to *ctsrvr.cfg*:

   ```
   SETENV DEBUG_JVM=S
   SETENV DEBUG_JVM_PORT=45987
   ```

   The first keyword the enables the c-treeACE SQL JVM debug feature by creating a TCP/IP socket at the port specified with the `DEBUG_JVM_PORT` keyword for a Java debugger to attach. In addition, the `DEBUG_JVM` keyword instruct the server to compile stored procedures with debugging information and to not remove the stored procedure source file from disk.
2. Start the c-treeACE SQL server.
3. Create a stored procedure (for example, “test”).
4. Examine the database directory (i.e. *ctreeSQL.dbs*) for a *.java* file. This is the Java file source. Pay particular attention to the class name (i.e. public final class admin_test_SP extends JavaBaseSP).
5. Start a Java debugger and attach it to localhost on the port specified by DEBUG_JVM_PORT.

   For example, using the Java debugger included with the JDK, run:

   ```
   jdb -attach 45987
   ```

   (run on the same machine running the server to access the Java source files).
6. Set a breakpoint on the method **dhSPwrap** of the stored procedure calls (i.e. admin_test_SP.dhSPwrap).
7. Call the stored procedure from any client side SQL tool such as ISQL.
8. The debugger should break at the start of the stored procedure.

2.2 JSWAT Example

JSwat is an open-source GUI Java debugger. It can be downloaded from https://github.com/nlfiedler/jswat (https://github.com/nlfiedler/jswat).

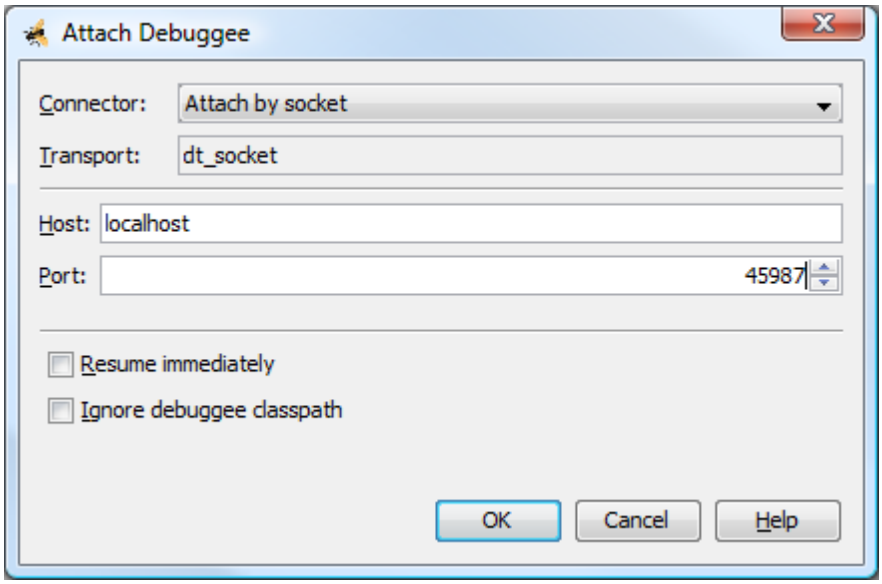
1. Add the following keywords to *ctsrvr.cfg*:

```
SETENV DEBUG_JVM=S
SETENV DEBUG_JVM_PORT=45987
```

2. Start the c-treeACE SQL Server.
3. Start ISQL and create a stored procedure as follows:

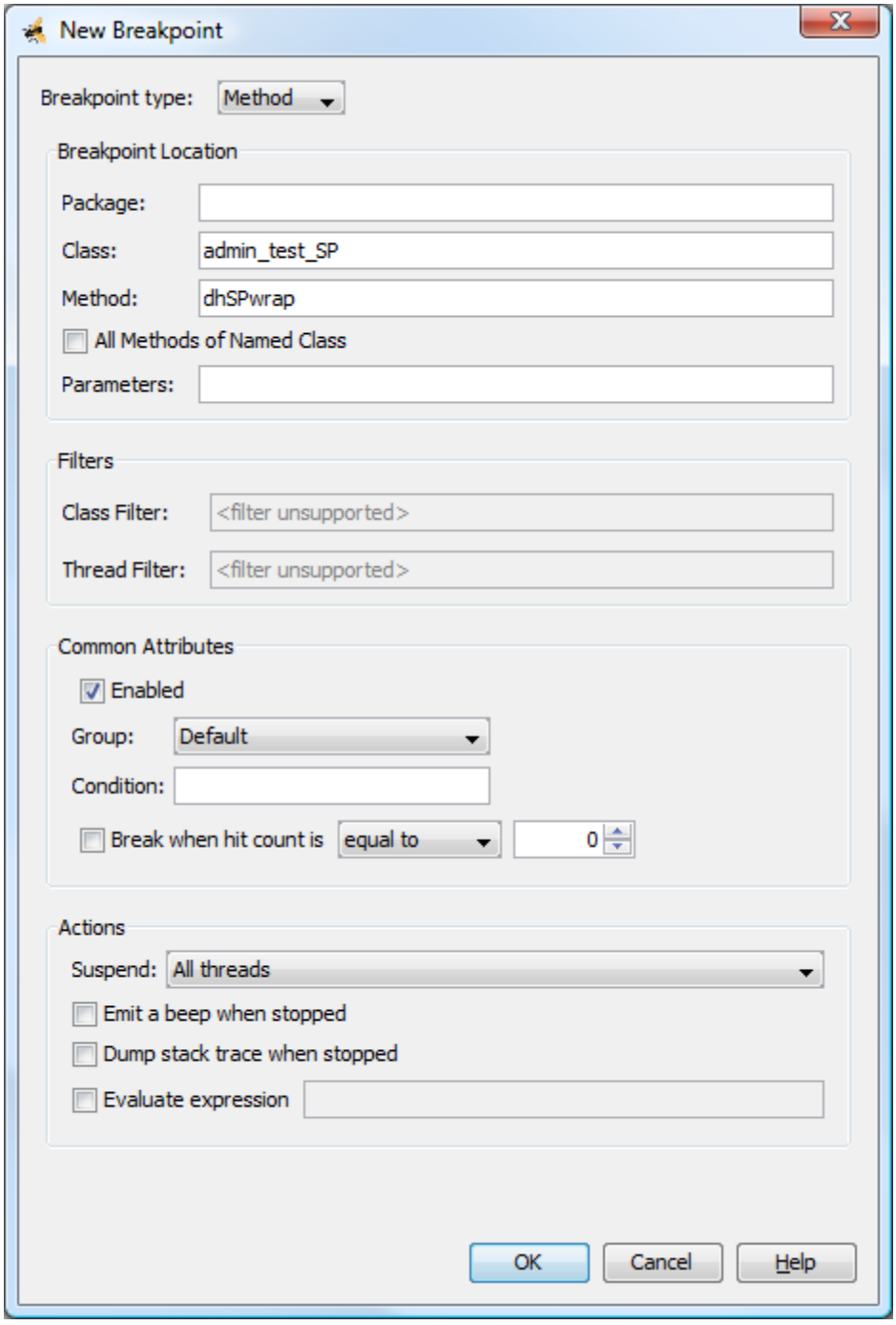
```
# isql -a ADMIN -u admin ctreeSQL
CREATE PROCEDURE admin.test( IN name CHAR (20) )
BEGIN
Integer testing = new Integer(24);
END
```

4. Start **jswat.**
5. Select “New Session” from the “Session” menu and edit the following information:
   a. Assign a name (for instance “c-treeACE Session”).
   b. Switch to the “classes” panel and add the path to the *ctreeSQLSP.jar* file and the path to the *ctreeSQL.dbs* directory.
   c. Switch to the “Sources” panel and add the path to the *ctreeSQL.dbs* directory.
6. Select the session just created from the Sessions Drop Down menu on the toolbar.
7. From the “Session” menu, click on “Attach”
8. Configure the Transport as "Attach by socket". Fill in Host and Port with information from your c-treeACE SQL Server configuration.



9. Click OK (The “debugger console” should now read” “VM attached to session c-treeACE Session”.
10. In the “Breakpoint” menu select “New Breakpoint” to create a new breakpoint.
    a. Set Breakpoint type to “Method.”
    b. Set Class to the stored procedure name (for example, admin_test_SP).

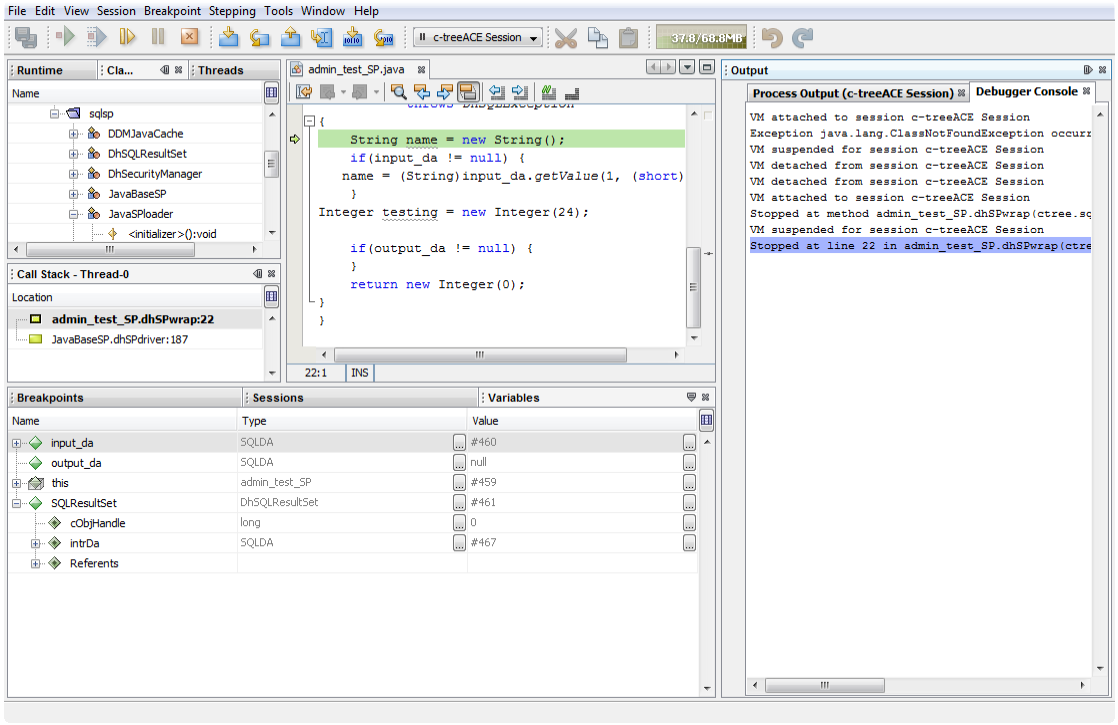
c. Set Method to “dhSPwrap.”



11. Return to ISQL and run the stored procedure:

```
call test ('John Smith');
```

12. The debugger displays the current line in the stored procedure.



3. Index