dbtester - DB Pool Testing sample application

{scrollbar}

Accessing application server specific features and using them in your J2EE application would make it more powerful than accessing only the J2EE features from them. It gives you the ability to write extensions to your application server.

This sample application lists all the database connection pools defined in Geronimo. You can select any of these connection pools and test the connectivity against the database as well as listing existing schemas and tables. In addition, the user can view the records of the each of the listed tables.

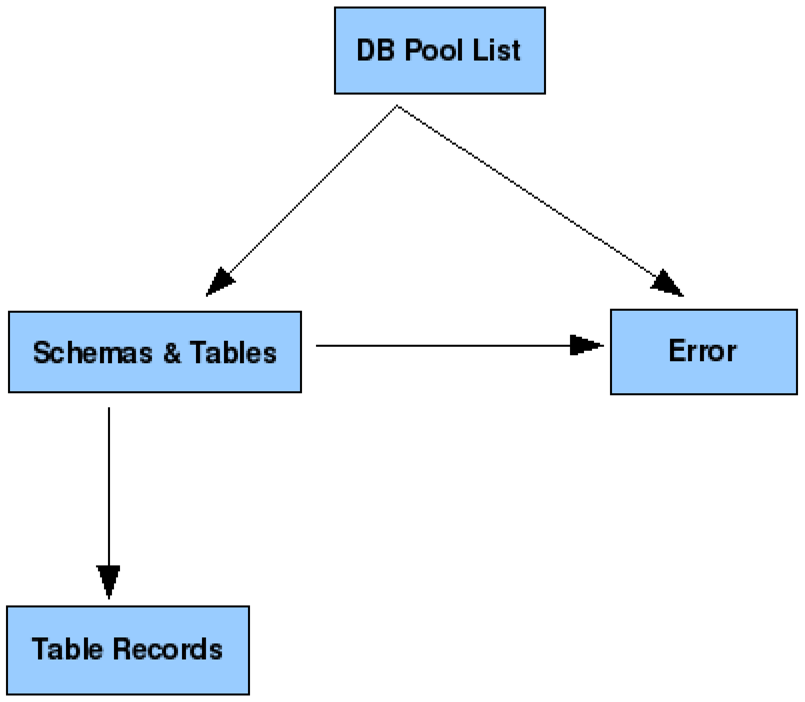
After reading this article you should be able to access Geronimo specific resources from your applications and use them in an effective manner.

This article is organized into the following sections :

Application overview

The sample application covered in this article will help you to test the Database connection pools deployed in your Geronimo server. One can consider this as an extension to the Geronimo console because the current version does not contain the ability to test connections to database pools after they have been deployed.

The following figure illustrates the application flow.



Welcome page of the application acts as a notice board which displays list of Database connection pools deployed in the Geronimo application server. Users can directly test those connection pools from the first page. If that particular connection pool requires a username and a password to get a connection, enter those details in the pop up window that appears. The list of database schemas and the tables associated with the connection pool will be displayed in the Schemas and Tables page. The contents of each table can be accessed from there on.

Application contents

The Inventory application consists of following list of packages.

- org.apache.geronimo.samples.dbtester.beans
  - DBManagerBean - Heart of the application which handles most of the application logic (including access of Geronimo Kernel).
- org.apache.geronimo.samples.dbtester.web
  - ContentTableServlet - Gets the content of a Database table and passes them to the presentation layer.
  - ListTablesServlet - Gets the list of schemas and tables associated with a database pool.

The list of web application files in the application is depicted by the following diagram.

```
#FFFFFF#FFFFFFsolid |- jsp +- common_error.jsp +- popup.jsp +- table_content.jsp +- table_list.jsp |- WEB-INF +- geronimo-web.xml +- web.xml |- index.
jsp
```

geronimo-web.xml defines the list of dependencies that have to be loaded into the web application class loader. In this case, there are no dependencies. Information about the project (e.g. module's unique identification, any dependencies) is described inside the <environment> tag. It is a good idea to give this module some sort of unique identification, so that it can later be referenced by some other deployable application. This module is in the group org. apache.geronimo.samples.dbtester. The path specified in the <context-root> tag will be the first segment of the URL used to access this web application. So to access this web application the url will be http://<hostname>:<port>/dbtester.

```
xmlgeronimo-web.xmlsolid <?xml version="1.0" encoding="UTF-8"?> <web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-2.0.1">
<environment> <moduleId> <groupId>org.apache.geronimo.samples</groupId> <artifactId>dbtester-war</artifactId> <version>2.1</version> <type>war<
/type> </moduleId> </environment> <context-root>/dbtester</context-root> </web-app>
```

The structure of the final WAR should look like the following:

```
#FFFFFF#FFFFFFsolid |- WEB-INF +- classes +- org +- apache +- geronimo +- samples +- dbtester |- web.xml |- geronimo-web.xml
```

web.xml defines two servlets that will act as the control layer between presentation and service layers.

```
xmlweb.xmlsolid <?xml version="1.0" encoding="UTF-8"?> <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
<description>dbtester Servlet Sample</description> <servlet> <display-name>ContentTableServlet</display-name> <servlet-name>ContentTableServlet<
/servlet-name> <servlet-class>org.apache.geronimo.samples.dbtester.web.ContentTableServlet</servlet-class> </servlet> <servlet> <display-
name>ListTablesServlet</display-name> <servlet-name>ListTablesServlet</servlet-name> <servlet-class>org.apache.geronimo.samples.dbtester.web.
ListTablesServlet</servlet-class> </servlet> <servlet-mapping> <servlet-name>ContentTableServlet</servlet-name> <url-pattern>/listContent</url-pattern>
</servlet-mapping> <servlet-mapping> <servlet-name>ListTablesServlet</servlet-name> <url-pattern>/listTables</url-pattern> </servlet-mapping>
<welcome-file-list> <welcome-file>index.html</welcome-file> </welcome-file-list> </web-app>
```

The most important part of this application is how to access Geronimo kernel and retrieve the list of database pools deployed there. This task is handled by the DBManagerBean class.

```
javaExcerpt from DBManagerBean.javasolid public DBManagerBean() { Kernel kernel = KernelRegistry.getSingleKernel(); Set<AbstractName> cfList =
kernel.listGBeans(new AbstractNameQuery(ResourceSource.class.getName())); for (AbstractName name : cfList) { try { Object rs = kernel.invoke(name,
"$getResource", new Object[]{}, new String[]{}); if (rs instanceof DataSource) { DataSource ds = (DataSource) rs; poolMap.put(name.getArtifact().
getArtifactId(), ds); } } catch (Exception e) { e.printStackTrace(); } } }
```

To retrieve the list of schemas and their tables, the application uses database metadata provided in a JDBC driver. ResultSet meta data has been used to get record related data and to display database contents. The following code snippet depicts how the application retrieves the schemas and their tables in a DataSource.

```
javaExcerpt from DBManagerBean.javasolid public Map getTableList(String poolName) throws SQLException { Map<String, List<String>> tableMap = new
HashMap<String, List<String>>(); if (poolMap.containsKey(poolName)) { DataSource ds = poolMap.get(poolName); Connection con = null; try { con = ds.
getConnection(); DatabaseMetaData metaData = con.getMetaData(); String[] tableTypes = {"TABLE"}; ResultSet rs = metaData.getTables(null, null, null,
tableTypes); while (rs.next()) { String schemaName = rs.getString("TABLE_SCHEM"); String tableName = rs.getString("TABLE_NAME"); List<String>
tableList; if (tableMap.containsKey(schemaName)) { tableList = tableMap.get(schemaName); tableList.add(tableName); } else { tableList = new
ArrayList<String>(); tableList.add(tableName); } tableMap.put(schemaName, tableList); } } finally { if (con != null) { try { con.close(); } catch (SQLException
e) { e.printStackTrace(); } } } } return tableMap; }
```

Usage

The app is available at http://localhost:8080/dbtester. Note that the sample servers you get if you run maven with -Pit will not show any data since they dont include any database pools. A more realistic view comes from installing the plugin or app on a full geronimo server.

Summary

This article has shown you how to access Geronimo related features from a J2EE application. You followed step-by-step instructions to build, deploy and test a sample application to elaborate these features. This sample application can be used as tester for database connection pools deployed in Geronimo.