RDM Server 8.3

ADO.NET Guide

Trademarks

Contents

Introduction to ADO.NET with RDM Server

ADO.NET provides a programming level interface for database access that is uniform and standard using the .NET Framework from Microsoft.

Creating a Database Application Using the RDM Server ADO.NET Data Provider

To develop an RDM Server database application, you must use Microsoft's .NET Framework 1.1 or later.

To create a database application program to access RDM Server, the following steps provide a general guideline.

1. Create a connection string identifying how to get to the server and what user name and password to connect with.
2. Connect to the server.
3. Create a statement handle.
4. Prepare and execute any SQL statements.
5. Get data results from the SQL execution.

Important: Please refer to the ADO.NET documentation before reading further in this document. This document assumes basic knowledge of ADO.NET interfaces and classes. A complete ADO.NET API reference is available on the MSDN website (http://msdn.microsoft.com/en-us/library/e80y5yhx.aspx).

RDM Server ADO.NET Connection String Syntax

The connection string used to connect to RDM Server via ADO.NET is basically a string that contains a set of key/value pairs. In particular the keys that are expected are "host", "port", "user", and "password".

A typical connection string would be "host=localhost; port=1530; user=admin; password=secret". The **host** key indicates the machine that the server is running on. This can be symbolic (i.e. the hostname of a machine) or dotted format (i.e. 192.168.0.1). If it is omitted the default value that will be used is 'localhost'. The **port** key indicates the port that the server will be listening. If this key is omitted then 1530 (which is the default listening port for the server) will be used. The **user** and **password** keys are required and specify the credentials for the logging into the server.

Example 1

```
RDMSConnection login(string user, string password)
{
    RDMSConnection conn;

    string connstr = "host=localhost; port=1530; user=" + user +
            "; password=" + password;

    conn = new RDMSConnection(connstr);
    conn.Open();
    conn.AutoCommit = true;

    return conn;
}
```

Creating a Statement Handle

Once you connect to the RDM Server database server successfully, the bulk of your database programming involves these tasks:

- executing SQL statements;
- processing data resulting from SQL query executions;
- error handling.

To process SQL statements, you must create an **RDMSCommand** class.

Example 2

```
public void prepareAndExecute(RDMSConnection conn)
{
    String ifile = "test_data.txt";
    String data;

    try {
        data = File.ReadAllText(ifile);
    }
    catch (Exception e) {
        Console.WriteLine("Could not read from file '" + ifile + "'");
        return;
    }

    try {
        // Create and prepare an SQL command
        RDMSCommand cmd = new RDMSCommand("insert into mytable " +
                "(timecol, datecol, chartext) values (?, ?, ?)", conn);
        cmd.Prepare();

        //Bind values
        RDMSParameter parm = new RDMSParameter("@parm1", DateTime.Now);
        parm.DbType = DbType.Time;
        cmd.Parameters.Add(parm);

        parm = new RDMSParameter("@parm2", DateTime.Today);
        parm.DbType = DbType.Date;
        cmd.Parameters.Add(parm);

        parm = new RDMSParameter("@parm3", data);
        cmd.Parameters.Add(parm);

        // execute the SQL statement
        cmd.ExecuteNonQuery();
```

```
            // if autocommit was off you could do a "conn.commit();"

            cmd._close();
        }
        catch (Exception e) {
            Console.WriteLine(e.ToString());
        }
    }
```

Getting Results of an SQL Execution

Resulting data (if any) from an SQL query is returned to the application via a **RDMSDataReader**.

Example 3

```
public void testReadBlobs(RDMSConnection conn)
{
    RDMSCommand    cmd;
    RDMSDataReader reader;

    try {
        //Create a SQL command
        cmd = new RDMSCommand("select timecol, datecol, chartext " +
                "from mytable", conn);

        // Execute the query
        reader = cmd.ExecuteReader();

        // Read each row and display the results
        while (reader.Read()) {
            Console.WriteLine(reader["timecol"]);
            Console.WriteLine(reader["datecol"]);
            Console.WriteLine("ASCII Data:");
            Console.WriteLine(reader["chartext"]);
        }

        cmd._close();
    }
    catch (Exception e) {
        Console.WriteLine(e.ToString());
    }
}
```