Dear Ada

David Botton
AdaPower
David@Botton.com
http://www.adapower.com

Dear Ada,

I need to get a handle on my problem, I've got this limited object see, and it needs to change. I am willing to give input, but I don't want it twisting my input. Help me get out of this in and out mess.

- Confused In a Limited Mess

A friend of mine let me in on his trick (the Rosen Trick from Jean-Pierre Rosen) for just this sort of problem It allows a limited object to be modified when passed as an in parameter thus treating the object more like a handle to an object.

```
with Ada.Text_IO; use Ada.Text_IO;

procedure Rosen_Trick is

   type T;

   type Relay_Type (Reference : access T) is limited null record;

   type T is limited
      record
         Relay : Relay_Type (T'Access);
         Data  : Natural := 0;
      end record;

   function Next (X : in T) return Integer is
      Data : Natural renames X.Relay.Reference.Data;
   begin
      Data := Data+1;
      return Data;
   end Next;

   Test_T : T;

begin

   for N in 1 .. 10 loop
      Put_Line(Integer'Image(Next(Test_T)));
   end loop;

end Rosen_Trick;
```

Dear Ada,

I am not much of a smoker, but some times I think a pipe can open communications. I don't have much experience how do I get a pipe going?

- Pipe Buddy

A quick way to get started with pipes is with a cross platform package called Pipe_Commands by Jim Rogers available at http://www.adapower.com/reuse/pipes.html.

Here is an example of using pipes to read in a directory listing:

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Strings.Unbounded; use Ada.Strings.Unbounded;

with Pipe_Commands; use Pipe_Commands;

procedure Pipe_Test is
   Pipe    : Stream;
   Command : constant String := "dir"; -- For Unix change to "ls"
begin
   Put_Line ("Results of command '" & Command & "'");
   Pipe := Execute (Command, Read_File);

   loop
      begin
       declare
          Buffer : String := To_String (Read_Next (Pipe));
       begin
          Put_Line (Buffer);
       end;
      exception
       when Pipe_Commands.End_Of_File =>
          exit;
      end;
   end loop;

   Close (Pipe);
end Pipe_Test;
```

Dear Ada,

The guys at work are always telling me that because of my language Bill won't give me Access to his records. If only I talked the same languages as nerd boy Bill, everything would be good.

- Stuck Behind the Language Barrier

No need to start talking Bill's favorite languages to talk to Access, SQL, or any other ODBC database or OLE DB data source.

Here is an example of reading records from an Access database using ADO with GNATCOM (http://www.adapower.com/gnatcom). The ADO package were generated with BindCOM (see the make.bat file in gnatcom\samples\bindings). The code for this example and the sample database is found in the GNATCOM installation's sample directory.

```
with GNAT.IO; use GNAT.IO;

with GNATCOM.Types;
with GNATCOM.BSTR; use GNATCOM.BSTR;
with GNATCOM.VARIANT; use GNATCOM.VARIANT;
with GNATCOM.Initialize;

with ADO.uConnection_Interface; use ADO.uConnection_Interface;
with ADO.uCommand_Interface; use ADO.uCommand_Interface;
with ADO.uRecordset_Interface; use ADO.uRecordset_Interface;
with ADO.Fields_Interface; use ADO.Fields_Interface;
with ADO.Field_Interface; use ADO.Field_Interface;

procedure ADO1 is
   use type GNATCOM.Types.VARIANT_BOOL;

   Connection : uConnection_Type;
   Command    : uCommand_Type;
   Recordset  : uRecordset_Type;
begin
   GNATCOM.Initialize.Initialize_COM;

   Put_Line ("Create ADO Engine");
   Create (Connection, ADO.CLSID_Connection);

   Put_Line ("ADO Version is : " & To_Ada (Get_Version (Connection)));
   New_Line;
   Put_Line ("Open Connection to Database");

   Put_ConnectionString
     (Connection,
      To_BSTR ("Provider=Microsoft.Jet.OLEDB.4.0; " &
               "Data Source=res\adotest.mdb"));

   Open (Connection, null, null, null, 0);

   Create (Command, ADO.CLSID_Command);
   PutRef_ActiveConnection (Command, Pointer (Connection));

   Put_CommandText (Command, To_BSTR ("SELECT * FROM People"));

   Create (Recordset, ADO.CLSID_Recordset);

   Open (Recordset,
         Source     => To_VARIANT_From_Dispinterface (Command),
         CursorType => ADO.AdOpenForwardOnly,
```

```
         LockType   => ADO.AdLockReadOnly,
         Options    => 0,
         Free       => False);

   MoveFirst (Recordset);

   while Get_EOF (Recordset) /= GNATCOM.Types.VARIANT_BOOL_TRUE loop
      declare
         Fields : Fields_Type;
      begin
         Attach (Fields, Get_Fields (Recordset));

         for N in 0 .. Integer (Get_Count (Fields)) - 1 loop
            declare
               Field : Field_Type;
            begin
               Attach (Field, Get_Item (Fields, To_VARIANT (N)));
               Put_Line (To_Ada (Get_Name (Field)) & " = " &
                         To_Ada (Get_Value (Field)));
            end;
         end loop;
      end;
      MoveNext (Recordset);
      New_Line;
   end loop;

   Put_Line ("Close Connections");
   Close (Recordset);
   Close (Connection);
end ADO1;
```

The ADO connection string used here is an OLE DB connection string, "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=res\adotest.mdb"; you can do OLE DB direct to SQL Server 7.0 using "Provider=SQLOLEDB.1; Persist Security Info=False; User ID=UserName; Initial Catalog=DatabaseName; Data Source=SQLServerName" and any configured ODBC DSN can be used, for example "DSN=MYDSN".

It is also possible to use a "UDL" file containing the connection information. To create a UDL file create a blank file with the extension ".UDL" then double click on the file to configure the data source information. Then use "File Name=MyUDLFile.udl" for the connection string for ADO.