Database Programming with Ada
Ada Europe 2010, Valencia

Frank Piron, frank.piron@konad.de
KonAd GmbH, In der Reis 5, D-79232 March-Buchheim
Content

• Why Ada for Database Programming?
• The Database Access Library
• The User Interface Library
• Projects
• Conclusion
Why Ada for Database Programming?

The Situation in 2001


- By the end of the millenium Oracle stopped further development of the Developer 2000 toolset.

- Oracle Forms does not allow 100% integration of system services.

- Oracle Forms modules are single threaded.

Begin of 2002

After 6 month of evaluation → Ada
Why Ada for Database Programming?

Pro

• Similarity between Ada and PL/SQL
• Features of Ada
  Multitasking
  Object Orientation
  System Integration
  Platform Independence
  Standardization
• Ada is a language for the development of big reliable software systems
Why Ada for Database Programming?

Contra

- Ada is not a mainstream language.
  - Are there Ada programmers?
  - What will the customers say?
  - Will Ada be available in 10 years?
- Ada is not easy to learn.
- There are only few libraries for Database Access and GUI-Programming.
Why Ada for Database Programming?

Start in 2002

Building knowledge of Ada95. Decision for GNAT.

We need libraries for Oracle Database Access and GUI-Programming on Windows.

Web search and evaluation

- Adaoci (Dmitriy Anisimkov) as a starting point for a database access library
- GWindows (David Botton) as a basis for the development of GUI-Components
The Database Access Library

Oracle

OCI

Adaoci

Konada.Db.Sql

.Rows

.Tables

Win32

Win32-API

GWIndows +

GWIndows_Extended

Other

Network

Controls (Single-, Multirow) Application
The Database Access Library

Why not e.g. GNADE for database access?

• We wanted to take full advantage of the Oracle Call Interface like
  • Asynchronous Execution
  • Blob Support
  • Advanced Queueing

• As few third party libraries as possible. Especially no ODBC driver.

• Ability to extend our library with new features of the Oracle Call Interface as soon as possible.
The Database Access Library

Konada.Db.Sql

Adaoci

Konada.Db

Sqltype

Methods

Create
Bind/Set
Execute
Fetch
Get
...

Management of Define an Bind Variables is Done by the Library
The Database Access Library

Konada.Db.Sql

declare
    Person: Sqltype;
begin
    -- Konada.Db.Sql
    Create(Person,
        "select * from emp where empno=:empno_to_find");
    -- Statement is prepared
    -- Bind variables are known now
    -- set :empno_to_find in sqltype-instance Person
    -- bind is implicit here
    Set(Sqlcmd  => Person,
        Position => 1,
        Value    => 7369);
    Execute(Person);
    Fetch(Person)
        declare
            Name: String:=Get(Person,"ename");
        begin
            Text_Io.Put_Line(Name);
        end
    end;
The Database Access Library

Rows and Tables

To get rid of Sql
declare
  Employees: Tabletype;
  Rows_Fetched: Natural := 0;
begin
  -- logon to the database
  Logon("scott/tiger@sun");
  -- create clientside table object
  -- for database table "emp"
  Create_From_Db_Table(Table => Employees,
                         Db_Table_Name => "emp");
  -- allow updates
  Grant_Access(Employees, Update);
  -- fill table with all rows
  Fetch_All(Employees, Rows_Fetched);
  -- set salary of the first two emps to 1500 $
  Set(Table => Employees, Name => "sal", Value => 1500.0);
  -- move to the next row
  Move(Table => Employees, To => Next);
  Set(Table => Employees, Name => "sal", Value => 1500.0);
  -- post changes to the database
  Post_Changes;
  -- and commit
  Commit;
end;
The Database Access Library
Konada.Db.Tables - Implementation

Double linked list

Data

Tagged Hierarchy

Data Element

Integer

Time

String

Current Row (Array)

Row Position

export
The User Interface Library

Oracle

OCI

Adaoci

Konada.Db.Sql

.Rows

.Tables

Network

Win32-API

GWindows +

GWindows_Extended

Other

Network

Controls (Single-, Multirow)
The User Interface Library

Goals

GUI – Components for several purposes:

• Display and manipulation of single and multiple Data Records

• Automatic and programmatic layout (no GUI-Builder)

• Flexible Event-Model

• Tree-Navigation with the tree reflecting the data model

• Storage of GUI-Layout into the database. The Modules adjust without recompilation
The User Interface Library

Why start with Gwindows?

GUI-centered thick binding to the win32-API by David Botton

✓ Easy to use
✓ Easy to extend
✓ Available under the GMGPL
The User Interface Library

Example: Multi Record Control

- List_View_Control
- Header-Drag-And-Drop Icons
- Header-Click-Event

GWInownds

Ex_List_View_Control

GWInownds_Extended

Konada.Db.Tables - Object
# The User Interface Library

## Multi Record Control

![User Interface Library Multi Record Control](image)

<table>
<thead>
<tr>
<th>SSTYPKURZBEZ</th>
<th>SSTYPBEZ</th>
<th>++ Methode</th>
<th>Controlled edit</th>
<th>Controlled edit</th>
<th>MEDIUMNR</th>
<th>IST_SYS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROEF</td>
<td>Restsaldo ROEF</td>
<td>TIFF</td>
<td>RE</td>
<td>RE</td>
<td>EDV</td>
<td></td>
</tr>
<tr>
<td>GA Zahlf.</td>
<td>GA Zahlungsunfähigkeit einford</td>
<td>TIFF</td>
<td>GA</td>
<td>GA</td>
<td>EDV</td>
<td></td>
</tr>
<tr>
<td>EenBetrag</td>
<td>hba26</td>
<td>TIFF</td>
<td>I-</td>
<td>I-</td>
<td>EDV</td>
<td></td>
</tr>
<tr>
<td>PfJab</td>
<td>hba12</td>
<td>TIFF</td>
<td>I-</td>
<td>I-</td>
<td>EDV</td>
<td></td>
</tr>
<tr>
<td>Schlussver</td>
<td>Schlussverfügung</td>
<td>TIFF</td>
<td>SC</td>
<td>SC</td>
<td>EDV</td>
<td></td>
</tr>
<tr>
<td>ProtAusk.</td>
<td>Protokoll Auskunftsperson</td>
<td>TIFF</td>
<td>PR</td>
<td>PR</td>
<td>EDV</td>
<td></td>
</tr>
<tr>
<td>Prot.Zeug.</td>
<td>Protokoll Zeuge/Zeugin</td>
<td>TIFF</td>
<td>PR</td>
<td>PR</td>
<td>EDV</td>
<td></td>
</tr>
<tr>
<td>Prot_Einspo</td>
<td>Protokoll Einsprecher/in</td>
<td>TIFF</td>
<td>PR</td>
<td>PR</td>
<td>EDV</td>
<td></td>
</tr>
<tr>
<td>Nachr_Urt.</td>
<td>Nachrichtung Unterschrift</td>
<td>TIFF</td>
<td>NA</td>
<td>NA</td>
<td>EDV</td>
<td></td>
</tr>
<tr>
<td>Rück_Post</td>
<td>Rücksendung Post</td>
<td>TIFF</td>
<td>RÜ</td>
<td>RÜ</td>
<td>EDV</td>
<td></td>
</tr>
<tr>
<td>Zv0_Private</td>
<td>Zv0_Private_Anzeige</td>
<td>TIFF</td>
<td>ZV</td>
<td>ZV</td>
<td>Papier</td>
<td></td>
</tr>
<tr>
<td>Bed.Arvr.</td>
<td>Berufung Antrag Abweisung BGZ</td>
<td>TIFF</td>
<td>BE</td>
<td>BE</td>
<td>EDV</td>
<td></td>
</tr>
<tr>
<td>Berufung R</td>
<td>Berufung Rückzug BGZ</td>
<td>TIFF</td>
<td>BE</td>
<td>BE</td>
<td>EDV</td>
<td></td>
</tr>
<tr>
<td>Berufung_A</td>
<td>Berufung Anmeldung BGZ</td>
<td>TIFF</td>
<td>BE</td>
<td>BE</td>
<td>EDV</td>
<td></td>
</tr>
<tr>
<td>Rechtskr.P</td>
<td>Rechtskr. RÜ</td>
<td>TIFF</td>
<td>RE</td>
<td>RE</td>
<td>EDV</td>
<td></td>
</tr>
<tr>
<td>Auftragsrpt.</td>
<td>Auftragsrpt. Dolmetscherin</td>
<td>TIFF</td>
<td>AU</td>
<td>AU</td>
<td>EDV</td>
<td></td>
</tr>
<tr>
<td>Rapport Nachtr.</td>
<td>Rapport Nachtrag</td>
<td>TIFF</td>
<td>RA</td>
<td>RA</td>
<td>EDV</td>
<td></td>
</tr>
<tr>
<td>Zv0_1ITA_NB_Firmen</td>
<td>Zv0_1ITA_NB_Firmen</td>
<td>TIFF</td>
<td>ZV</td>
<td>ZV</td>
<td>FTW</td>
<td></td>
</tr>
</tbody>
</table>

Datensatz 9/113
The User Interface Library

Multi Record Control and Data Container

List View Control

Konada.Db.Tables

Row Position

rowid
begin
   -- create sr_control on database table <emp>
   Single_Row.Create(Control => Sr_Control,
                      Parent => Main_Window,
                      Query => "select rowid, e.* from emp e",
                      Connect_String => "scott/tiger@tut");

   -- grant update access (read access is always granted implicitly)
   Single_Row.Grant_Access(Sr_Control, Konada.Db.Tables.Update);

   -- fill data
   Single_Row.Fill(Control => Sr_Control, Rows_To_Fetch => -1, -- all rows
                    Rows_Fetched => Rows_Fetched);

   -- move the internal row pointer to the first data row
   Single_Row.Move(Control => Sr_Control, Move_To => First,
                    Success => Success, Info => Info);
end $rкт1;
The User Interface Library

The Effect of the previous code snippet
Projects

- ElSch – Workflow Client
- Complete ERP-Solution for hydraulic and pneumatic element manufacturers (Demo)
- Interface between Workflow System and Archive System on the Solaris platform
- Further development of our dynamic GUI control
Projects - EISch

EISch – Workflow Client. Running since 2005
Stadtrichteramt Zürich
# Projects – ERP Solution

## Partner

<table>
<thead>
<tr>
<th>KontoNr</th>
<th>Firmenname</th>
<th>Kurzname</th>
<th>Name</th>
<th>Name 2</th>
<th>Name 3</th>
<th>Aktiv</th>
<th>Branche</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>300164</td>
<td>Kunde</td>
<td>Bochito</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>300163</td>
<td>Kunde</td>
<td>Girde</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>700138</td>
<td>Vertreter</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Anschriften

<table>
<thead>
<tr>
<th>Name</th>
<th>Liebherr KG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name 2</td>
<td>Maschinenbau</td>
</tr>
<tr>
<td>Name 3</td>
<td></td>
</tr>
<tr>
<td>Straße</td>
<td>Mühlengraben</td>
</tr>
<tr>
<td>Hausnr.</td>
<td>11</td>
</tr>
<tr>
<td>Postlech</td>
<td></td>
</tr>
</tbody>
</table>

## Kontakte

<table>
<thead>
<tr>
<th>Anrede</th>
<th>Herr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vorname</td>
<td>Anton</td>
</tr>
<tr>
<td>Name</td>
<td>Huber</td>
</tr>
<tr>
<td>E-Mail</td>
<td></td>
</tr>
<tr>
<td>Telefon</td>
<td>0758 676786</td>
</tr>
</tbody>
</table>

## Vorsicht

Verpackt in Karten vollständig

Sprache: Deutsch

Aktiv: 

Vertreter: 38

MWST: 19%

Währung: Euro

angelegt von: HYDRAULKA

geändert am: 07.03.2007
Projects
Dynamic GUI Control

- DB
- GUI Layout
- Application

GUI - Maintenance
Database Programming with Ada is real business, but

• We had to learn and turn

• The customers have to trust our choice

• We often have to give answers to the question:

„Why not Java?“

Here is one:

„We like programming in Ada.“