6.07

www.dotnet-magazin.de

S&S

dot.net magazin

.NET, Visual Studio & More

dot.net magazin

with CD →

.NET Security

Thomas Caspers
Interview
Wie sicher ist die WCF?

■ SQL Server 2005 – extra data security

■ CAS-Sandbox – Security wit

Team

Special Edition

fecher.
good people
good software

rwaltung
gestern

Continuous Integration with VSTS
Power boost thanks to NGen 2.0

More speed for your application
Recover a wealth of data

Data mining with SQL server
2005 analysis services

CD-content: trial versions, .NET tools & more

- Advantage Database Server 8.1
- Allora V5.0.3
- Doc-to-Help (preview)
- FlyGrid.NET 1.5
- novaPDF SDK 5.0
- Vanatec OpenAccess
- Scissors Mobile 2007
- SmartInspect 2.1.1
- TeamCity 2.0
- TestAdvantage 2007 Vol.1
- TurboDB Managed
- ...

Latecomer from the 90s

Migrating SQLWindows applications to .NET with the aid of the Ice Tea Group's porting tools

by Johann Baumeister

Since the 1990s, Gupta SQLWindows has been used to develop many applications that were deployed primarily in small and medium-sized firms. To ensure not only its future deployment but also its continued development, the Ice Tea Group, a group of SQLWindows developers, provides comprehensive porting options to the .NET Framework 2.0.

The 1990s' equivalent of today's Visual Studio was SQLWindows – the principle development tool for advanced Windows programming, especially of business applications for small and medium-sized firms.

Many of the applications created at that time are still in use and would be difficult to replace. The programming model for those Windows applications would nowadays be described as fat (or rich) client model. The SAL programming language (SQLWindows Application Language) is proprietary, as are most 4GL languages.

The "fourth generation languages" (4GL) were described as those languages that were developed to take over from the previous traditional languages such as C, C++ or Smalltalk. However, SQLWindows was not the only 4GL. At that time, the other members of the "Gang Of Four" besides Gupta were Borland's Delphi, Powerbuilder and Progress 4GL. The uniqueness of 4GLs, and SQLWindows in particular, was the way in which SQL was directly integrated. This principle could certainly be compared to the current concept adopted by Microsoft with its LINQ (Language Integrated Query). LINQ integrates SQL, XLink and XQuery queries directly into the source code. Unlike the SQL queries used by SQLWindows, LINQ statements are precompiled and then integrated as part of the code instead of being integrated as SQL commands. Precompilation of the statements allows the compiler to run a syntax check. This prevents attacks such as those using SQL injections.

SQLWindows and the other 4GL products of that era were not only the leading development tools of the time but above all were the main players in the client / server movement that gained momentum in those days. These tools enabled business applications for the PC to be developed for the first time on such a large scale. According to Gupta, approximately 10,000 licenses of the development environment were sold. .SQLWindows was used in a wide variety of business sectors. The applications range from telecommunications, the financial sector and medicine as well as mechanical engineering.

clear & brief

Contents

An overview of SQLWindows application to .NET migration

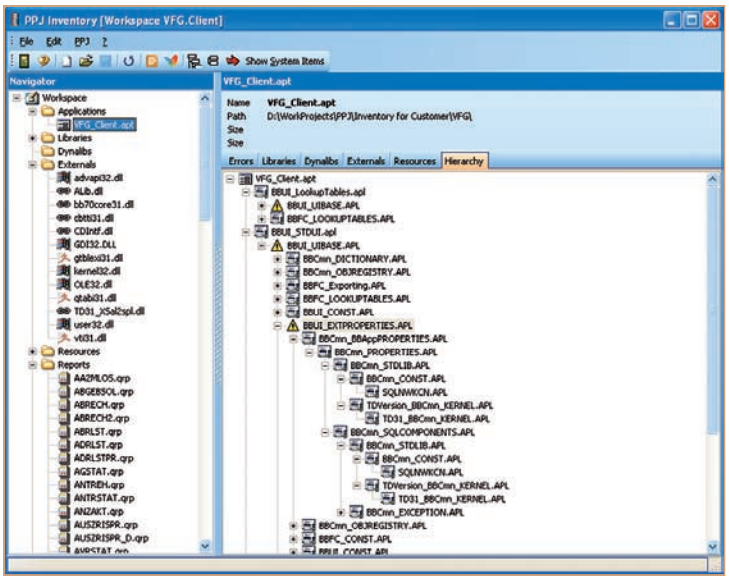
Summing up

Many applications for small and medium-sized firms were created in the 1990s using SQLWindows. Now these applications can finally be migrated to .NET thanks to tools from the Ice Tea developer group

Back to the future with Ice Porter

IT would not be IT if it were not capable of self-replication. SQLWindows developers would like to be able to use translation technology to compile their source code into different code forms.

Every compiler and runtime engine that processes intermediate code is after all nothing more than a translator that takes the code of a specific language and syntax and immediately transforms it into a different language and syntax, while retaining the same semantics. When you consider this fact, it is surprising that this technique is not used more often. Of course, exceptions are the constructs and features peculiar to each language. Every second or third generation programming language, which implies Java, Visual Basic, C#, pseudo-code and even the latest concepts such as XAML, are basically just programming metaforms and are always being transformed from one form into another.


PPJ Inventory [Workspace VFG.Client]
File Edit PPJ ?
Show System Items
Navigator
Errors Libraries Dynalibs Externals Resources Hierarchy

```
Workspace
  Applications
    VFG_Client.apt
  Libraries
  Dynalibs
  Externals
    advapi32.dll
    ALib.dll
    bb70core31.dll
    cbtti31.dll
    CDIntf.dll
    GDI32.DLL
    gtblexi31.dll
    kernel32.dll
    OLE32.dll
    qtabi31.dll
    TD31_XSal2spl.dll
    user32.dll
    vti31.dll
  Resources
  Reports
    AA2MLOS.qrp
    ABGEBSOL.qrp
    ABRECH.qrp
    ABRECH2.qrp
    ABRLST.qrp
    ADRLST.qrp
    ADRLSTPR.qrp
    AGSTAT.qrp
    ANTREH.qrp
    ANTRSTAT.qrp
    ANZAKT.qrp
    AUSZRISPR.qrp
    AUSZRISPR_D.qrp
    AVRSTAT.qrp
```

```
VFG_Client.apt
Name  VFG_Client.apt
Path  D:\WorkProjects\PPJ\Inventory for Customer\VFG\
Size
Size

VFG_Client.apt
  BBUI_LookupTables.apl
    BBUI_UIBASE.APL
    BBFC_LOOKUPTABLES.APL
  BBUI_STDUI.apl
    BBUI_UIBASE.APL
      BBCmn_DICTIONARY.APL
      BBCmn_OBJREGISTRY.APL
      BBFC_Exporting.APL
      BBFC_LOOKUPTABLES.APL
      BBUI_CONST.APL
      BBUI_EXTPROPERTIES.APL
        BBCmn_BBAppPROPERTIES.APL
          BBCmn_PROPERTIES.APL
            BBCmn_STDLIB.APL
              BBCmn_CONST.APL
                SQLNWKCN.APL
              TDVersion_BBCmn_KERNEL.APL
                TD31_BBCmn_KERNEL.APL
            BBCmn_SQLCOMPONENTS.APL
              BBCmn_STDLIB.APL
                BBCmn_CONST.APL
                  SQLNWKCN.APL
                TDVersion_BBCmn_KERNEL.APL
                  TD31_BBCmn_KERNEL.APL
              BBCmn_EXCEPTION.APL
        BBCmn_OBJREGISTRY.APL
        BBFC_CONST.APL
        BBUI_CONST.APL
```

Fig. 1: The source program is analyzed using PPJ-Inventory. The values provide a ballpark time estimate for the upcoming porting

A group of SQLWindows developers appears to be of the same opinion. This group provides a SQLWindows to .NET code translator called *Ice Porter*.

These programmers, who joined forces under the name *Ice Tea Group* [1], have set themselves the "Porting Project" challenge of transforming the remaining applications as simply as possible into .NET code using the *Ice Porter*. A compelling reason to work on this project was the takeover of *Gupta* by the *Unify Corporation*. Some software houses were concerned about the continued development and future of Gupta tools, which in the meantime are being sold as Centura tools. This is not without justification, since this is already the third takeover after the change of ownership from *Gupta* to *Platinum Equity* in 2000 and then to *Halo Technology Holdings, Inc* at the beginning of 2005. The Hessian software company *fecher*, which is also a member of the porting project, has supported and successfully implemented many Gupta projects in German-speaking markets. These markets have always been of particular importance to Gupta[2]. *Fecher* also consider the outlook for *Centura Team Developer* as a Windows development platform to be at risk and will be giving priority to .NET in future.

Porting a program from SQLWindows to C# or Visual Basic involves several stages. Firstly, the SQLWindows code is analyzed in order to estimate the effort involved in porting the application. The Porting Project supplies an inventory tool, *PPJ Inventory*, for this purpose. This analyses the complexity of the code, for example, the number of lines of code, the functions, windows or external libraries. The tool is provided free of charge and can be obtained, for example, from *fecher*. The results of the initial analysis provide a ballpark figure for the effort needed for the upcoming porting.

Translating code using the Ice Porter

The code translation itself is done using the *Ice Porter*. The SQLWindows source program is passed to the *Ice Porter*, which

Listing 1

A sample procedure in SQLWindows

```
Function: GotCompanyID
 Description: Uses the the COMPANY_ID table to generate
                                          a new company ID
    value.
    GotCompanyID( dfnID )
    Returns
Boolean:
    Parameters
     Receive Number: nID
    Static Variables
      Local variables
Number: n
      Actions
       If SqlPrepareAndExecute ( hSql,
                                   'SELECT @NULLVALUE
           (MAX(COMPANY_ID), 0) INTO :nID FROM
                                            COMPANY' )
        If SqlFetchNext( hSql, n )
         Set nID = nID + 1
         Return TRUE
       Return FALSE
```

then generates .NET code as best as it can in the chosen language. Visual Studio processes it further and compiles it. This will always take the form of a project, i.e. it will span several months and also require manual intervention. It is virtually impossible to perform a totally automatic transfer without developer involvement. QuickObjects from SQLWindows *QuickGraph*, *QuickMail* or *QuickReports* are not supported. Alternative approaches in .NET must be implemented. In each case, the effort involved in porting is substantially lower than the cost of a brand new development. Of course, it ultimately depends on the nature of each individual project. The *Ice-Porter* does not claim to be able to port everything. The smaller market and sales figures compared to those of a universal development tool also account for more restricted development efforts for the *Ice Porter*. After porting the source code with the *Ice Porter*, code optimization (refactoring) can begin. According to *fecher*, part of the manual process is the adaptation of low-level system routines. These have already been found by the initial code analysis and any manual corrections are performed by the porting team. ActiveX controls and COM objects usually run without any problem, since we are dealing with a Microsoft technology that is also supported in the target environment.

The database can be migrated while the source code is being converted. At a very early stage *Gupta* supplied its own database, *SQLBase*, for data storage. It is also still being maintained and developed but is now trailing behind the competition, i.e. the big players such as IBM, Oracle und Microsoft. These days it can still be found, especially in the niche market of embedded systems. The decline is unlikely to be due to the technology used by SQLbase, but rather due to the fact that the competition offers the entry-level version of their databases in the form of a free express edition. It is very difficult for a database that is not free to maintain its position against such odds.

SQLWindows lives on in spite of .NET

The Ice Porter provided by members and partners of the Porting Project is without a doubt a supportive and useful tool for porting SQLWindows applications. Nevertheless, the Unify Corporation does not intend to drop its SQLWindows and has recently released an update of that once highly popular development tool in the form of Team Developer 5.0. In any event, it remains an exciting time for the developers involved – perhaps too much excitement at times.

Johann Baumeister is a University educated computer scientist with many years of experience in the development, application and rollout of IT systems. His specialties include Web applications, collaboration, client / server and databases. He lives near Munich and works as a freelance consultant and technical author.

Eternal Links & References

[1] www.iceteagroup.com
[2] www.fecher.eu und www.fecher.eu/porting

Listing 2

The conversion result in C#

```
/// <summary>
/// Uses the the COMPANY_ID table to generate a new
                                              company ID
/// value.
///
/// GotCompanyID( dfnID )
/// </summary>
/// <param name="nID"></param>
/// <returns></returns>
public static SalBoolean GotCompanyID
                                        (ref SalNumber nID)
{
 #region Local Variables
 SqlLocals.GotCompanyID locals = new SqlLocals.Got
                                             CompanyID();
 #endregion
 #region Actions
 using (new SqlContext(locals))
{
  try
  {

 // PPJ: Assign parameters to the locals instance.
  locals.nID = nID;
  if (Var.hSql.PrepareAndExecute("SELECT @
       NULLVALUE(MAX(COMPANY_ID), 0)INTO :nID FROM
                                           COMPANY"))
  {
   if (Var.hSql.FetchNext(ref locals.n))
   {
    locals.nID = locals.nID + 1;
    return true;
   }
 }
 return false;
  }
  finally
  {

  // PPJ: Assign back receive parameters.
  nID = locals.nID;

  }
 }
 #endregion
}
```

fecher.

good people
good software

fecher e. Kfm.
Eberhard Fecher
Seestrasse 2-4
63110 Rodgau
Tel: +49-6106 605-0
Fax: +49-6106-605-200
E-mail:
eberhard.fecher@fecher.eu
Internet: www.fecher.eu