

dot.net

magazin

.NET, Visual Studio & More

.NET Security



- SQL Server 2005 – mehr Sicherheit für Daten
- CAS-Sandbox – Sicherheit ohne ...

Team gew...

Ein Sonderdruck der
Firma Fecher



Integration mit VSTS

Leistungsschub dank NGen 2.0

Mehr Speed für Ihre Anwendungen

Datenschätze bergen

Data Mining mit den SQL Server
2005 Analysis Services

CD-Inhalt: Testversionen, .NET Tools & More

- Advantage Database Server 8.1
- Allora V5.0.3
- Doc-to-Help (Vorschau)
- 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
- u.v.a.m.

Alle Infos ab S. 53

webinale⁰⁷
development design business

Datenträger enthält nur
Lehr- oder Infoprogramme



0.6

Nachzügler aus den 90ern

Die Migration von SQLWindows-Anwendungen nach .NET mithilfe der Portierungstools der IceTeaGroup

von Johann Baumeister

Mit Gupta SQLWindows wurden seit den 90er Jahren viele Anwendungen entwickelt, die vor allem im Mittelstand zum Einsatz kommen. Um diese auch in Zukunft nicht nur einsetzen, sondern auch weiterentwickeln zu können, bietet die IceTeaGroup, eine Gruppe von SQLWindows-Entwicklern, eine umfassende Portierungsmöglichkeit auf das .NET Framework 2.0.

Was Visual Studio heute ist, das war SQLWindows in den 90er Jahren des vergangenen Jahrhunderts – das zentrale Entwicklungswerkzeug für fortschrittliche Windows-Programmierung, insbesondere für kaufmännische Anwendungen für den Mittelstand. Dementsprechend sind viele der damals entstandenen Anwendungen heute noch im Einsatz und alles andere als leicht zu ersetzen. Vom Programmiermodell her betrachtet waren es Windows-Applikationen, die heutzutage als Rich Client oder Fat Client bezeichnet werden

würden. Die Programmiersprache SAL (*SQLWindows Application Language*) ist proprietär wie praktisch alle 4GL-Sprachen. Als „Sprachen der vierten Generation“ (4GL) wurden jene Sprachen bezeichnet, die als Nachfolger der bis dahin traditionellen Sprachen wie etwa C, C++ oder Smalltalk entwickelt wurden. SQLWindows stand jedoch nicht allein. Die „Gang Of Four“ bildeten seinerzeit neben Gupta auch Borlands Delphi, Powerbuilder und Progress 4GL. Das Besondere an den 4GLs und vor allem an SQLWindows war die direkte Integration von SQL. Das Prinzip war durchaus vergleichbar mit den Konzepten die Microsoft heute mit LINQ (*Language INtegrated Query*) verfolgt. Bei LINQ werden SQL-, XLink- und XQuery-Anfragen direkt in den Quelltext eingebunden. Im Unterschied zu den SQL-Abfragen von SQLWindows werden die LINQ-Statements jedoch bereits durch den Compiler umgesetzt und dann als Codebausteine und nicht als SQL-Kommandos integriert. Durch die vorherige Übersetzung der

Statements durch den Compiler kann dieser einen Syntaxcheck durchführen. Angriffe wie etwa durch SQL-Injections werden somit verhindert.

SQLWindows und auch die anderen 4GL-Produkte dieser Zeit waren aber nicht nur die führenden Entwicklungswerkzeuge jener Jahre, sondern sie waren auch beziehungsweise in erster Linie die Protagonisten der zu dieser Zeit immer stärker aufkommenden Client/Server-Bewegung. Diese Werkzeuge ermöglichten erstmals die großflächige Entwicklung von Geschäftsapplikationen für den PC. Nach Angaben von Gupta wurden in etwa 10.000 Lizenzen der Entwicklungsumgebung verkauft. Eingesetzt wurde SQLWindows in den unterschiedlichsten Branchen. Die Anwendungen finden sich sowohl in der Telekommunikation, in der Finanzwelt und der Medizin als auch im Maschinenbau.

Zurück in die Zukunft mit IcePorter

Die IT wäre nicht die IT, wenn sie sich nicht selbst reproduzieren könnte.

kurz & bündig

Inhalt

Ein Überblick über die Migration von SQLWindows-Anwendungen nach .NET

Zusammenfassung

Mit SQLWindows wurden in den 90er Jahren viele Applikationen für den Mittelstand erstellt. Dank der Tools der Entwicklervereinigung *IceTea Group* können diese Anwendungen endlich nach .NET migriert werden

Durch „Translator“ wollen SQLWindows-Entwickler ihren Quellcode in andere Codeformen übersetzen können. Wenn man bedenkt, dass jeder Compiler und auch die Runtime-Engines, die Zwischencode verarbeiten, letztendlich nichts anderes sind als Übersetzer, die einen Code einer bestimmten Sprache und Syntax nehmen und diesen flugs in eine anderen Sprache mit deren Syntax, bei gleichbleibender Semantik transformieren, ist es eigentlich schon etwas verwunderlich, dass diese Technik nicht noch öfter angewandt wird. Ausgenommen davon sind natürlich die jeweils spezifischen Konstrukte und Eigenheiten der Sprache. Aber jede Programmiersprache der zweiten oder dritten Generation und damit Java, Visual Basic oder auch C#, auch die Pseudocodes und sogar neueste Konzepte wie beispielsweise XAML sind letztendlich lediglich Metaformen der Programmierung und werden beständig von einer Form in eine andere transferiert.

So sah es wohl auch eine Gruppe von SQLWindows-Entwicklern. Unter dem Namen *IcePorter* bietet diese Gruppe einen Übersetzer von SQLWindows in .NET-Code an. Diese Programmierer, die sich unter dem Namen *IceTeaGroup* zusammenschlossen [1], haben sich mit dem „Porting Project“ zur Aufgabe gesetzt, die noch verbleibenden Anwendungen durch den *IcePorter* möglichst einfach in .NET-Code zu transferieren. Eine besondere Motivation erhielt die Arbeit an diesem Projekt durch die Übernahme von *Gupta* durch die *Unify Corporation*, denn so manches Software-Haus sah die Weiterentwicklung und Zukunft der Gupta-Tools, die zwischenzeitlich als Centura-Tools vermarktet wurden, mit Sorge. Dies ist nicht ganz unbegründet, da diese Übernahme nach dem Wechsel von *Gupta* zu *Platinum Equity* im Jahr 2000 und dann zu *Halo Technology Holdings, Inc.* Anfang 2005 schon die dritte Übernahme ist. Im deutschsprachigen Raum, der von jeher eine bedeutende Rolle für Gupta spielte, hat das hessische Software-Unternehmen *fecher*, ebenfalls Mitglied beim Porting Project, viele Gupta-Projekte begleitet und erfolgreich umgesetzt [2]. Auch *fecher* hält die Zukunft des *Centura Team Developer* als Windows-Entwicklungsplattform gefährdet und setzt für die Zukunft konsequent auf .NET.

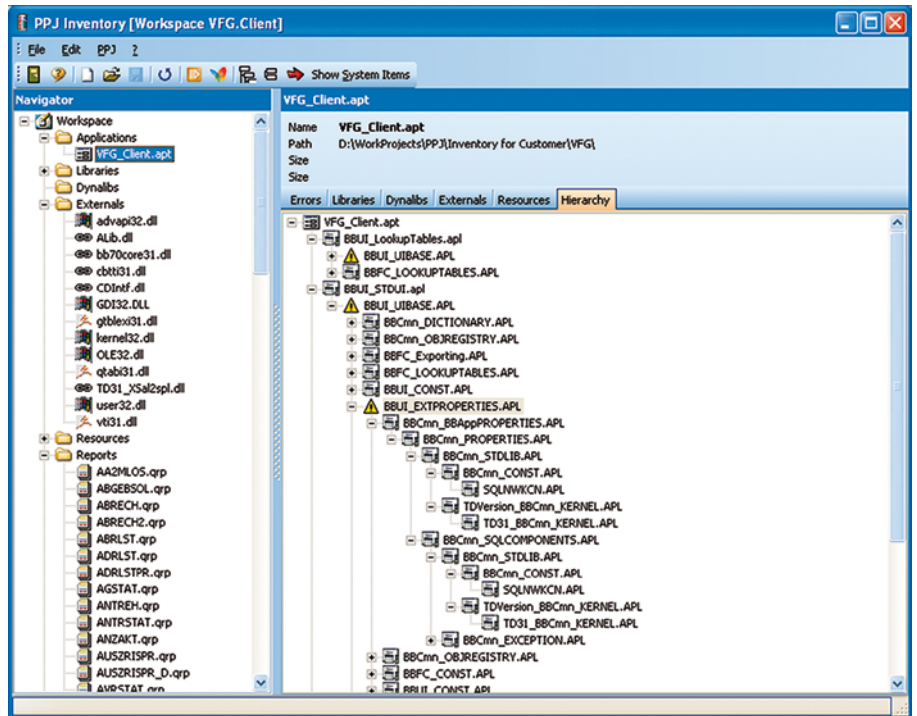


Abb. 1: Durch das PPJ-Inventory erfolgt eine Analyse des Quellprogramms. Die Werte dienen als Richtwert für Zeitaufwand der anstehenden Portierung

Das Tool wird frei zur Verfügung gestellt und kann zum Beispiel über *fecher* bezogen werden. Das Ergebnis dieser ersten Analyse dient als Richtwert für den anstehenden Portierungsaufwand.

Die Portierung eines Programms von SQLWindows nach C# oder Visual Basic durchläuft mehrere Stufen. Um den Aufwand für die Portierung der Applikation abschätzen zu können, erfolgt vorher eine Analyse des SQLWindows-Codes. Dazu liefern die Mitarbeiter des Porting Projects ein Inventarisierungstool, das *PPJ Inventory*. Dieses analysiert die Komplexität des Quellcodes, wie etwa die Anzahl der Codezeilen, der Funktionen, der Fenster oder der externen Bibliotheken.

Das Tool wird frei zur Verfügung gestellt und kann zum Beispiel über *fecher* bezogen werden. Das Ergebnis dieser ersten Analyse dient als Richtwert für den anstehenden Portierungsaufwand.

Codeumsetzung durch den IcePorter

Die eigentliche Codeumsetzung geschieht durch *IcePorter*. Diesem übergibt man die Quellprogramme der SQLWindows-Anwendung. Der *IcePorter* generiert daraus, so gut er kann, .NET-Code in der gewählten Zielsprache. Dieser wird in Visual Studio weiterbearbeitet und kompiliert. In jedem Fall wird es

Listing 1

Eine Beispielprozedur in SQLWindows

Function: GotCompanyID
 Description: Uses the the COMPANY_ID table to generate a new company ID
 value.
 GotCompanyID(dfn ID)
 Returns
 Boolean:
 Parameters
 Receive Number: nID
 Static Variables

Local variables
 Number: n
 Actions
 IfSqlPrepareAndExecute (hSql,
 'SELECT @NULLVALUE
 (MAX(COMPANY_ID), 0)INTO :nID FROM
 COMPANY')
 IfSqlFetchNext(hSql, n)
 Set nID = nID + 1
 Return TRUE
 Return FALSE

sich immer um ein Projekt handeln, d.h. es wird sich über einige Monate erstrecken und auch den manuellen Eingriff erfordern. Eine vollautomatische Übertragung ohne Zutun der Entwickler wird kaum machbar sein. Nicht unterstützt werden die Quick-Objekte von SQL-Windows *QuickGraph*, *QuickMail* oder *QuickReports*. Hier müssen alternative Lösungen in .NET realisiert werden. Verglichen mit einer Neuentwicklung ist der Aufwand für die Portierung in jedem Fall deutlich niedriger anzusetzen. Letztendlich hängt es aber natürlich immer vom individuellen Projekt ab. *IcePorter* erhebt nicht den Anspruch, alles portieren zu können. Im Gegensatz zu einem universellen Entwicklungswerkzeug ist der Absatzmarkt und damit die zu erlösenden Umsätze natürlich weitaus geringer und erlauben daher auch nur einen begrenzten Entwicklungsaufwand für den *IcePorter*. Nach der Portierung des Quellcodes durch den *IcePorter* kann die Optimierung des Codes (Refactoring) erfolgen. Zuden manuellen Arbeiten gehört laut *fecher* die Anpassung bei sehr systemnahen Routinen. Diese werden bereits bei der anfänglichen Code-Analyse aufgedeckt, die manuellen Nacharbeiten

erledigt das Portierungs-team. ActiveX-Controls und COM-Objekte laufen in aller Regel problemlos, da es sich hier um eine Technologie von Microsoft handelt, die auch in der Zielumgebung unterstützt wird.

Parallel zur Konvertierung des Quellcodes kann auch eine Umstellung der Datenbank erfolgen. Zur Datenhaltung lieferte *Gupta* sehr früh schon die eigene Datenbank *SQLBase*. Auch sie wurde bis heute gepflegt und weiterentwickelt, ist aber in Konkurrenz zu den „Big Playern“, wie etwa IBM, Oracle und Microsoft immer weiter zurückgefallen. Heute ist sie vor allem im Nischenmarkt der Embedded-Systeme anzutreffen. Die Ursachen für den Rückgang sind aber kaum der Technik der *SQLBase* anzulasten, sondern liegen nicht zuletzt in der Tatsache begründet, dass die Mitbewerber die Einstiegsversionen ihrer Datenbanken frei als Express-Editionen abgeben. Dagegen kann sich eine nicht kostenfreie Datenbank nur sehr schwer behaupten.

SQLWindows lebt trotz .NET

Die Mitglieder und Partner des Porting Project liefern mit dem *IcePorter* zweifelsfrei ein hilfreiches und nützliches Werkzeug zur Portierung von SQLWin-

dows-Anwendungen. Dennoch beabsichtigt die Unify Corporation nicht, ihr *SQLWindows* aufzugeben und hat jüngst mit dem *Team Developer 5.0* ein aktualisiertes Release des einst so beliebten Entwicklungswerkzeugs veröffentlicht. Für die involvierten Entwickler bleibt es in jedem Fall spannend – manchmal vielleicht mehr als gewünscht.

Johann Baumeister besitzt als studierter Informatiker langjährige Erfahrung in der Entwicklung, Anwendung und dem Rollout von IT-Systemen. Zu seinen Schwerpunkten zählen Webanwendungen, Collaboration, Client/Server und Datenbanken. Er lebt im Landkreis München und arbeitet als freier Berater und Fachautor.

● Links & Literatur

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

Listing 2

Das Ergebnis der Konvertierung in C#

```

/// <summary>
/// Uses the the COMPANY_ID table to generate a new
/// 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
        {
            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
        }
    }
    // PPJ: Assign parameters to the locals instance.
  
```



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