Bekannt und immer wieder neu: Upgrade Forms 6i auf Forms 11g Release 2

Perry Pakull
Trivadis AG
Glattbrugg

Schlüsselworte
Forms, Reports, Upgrade.

Einleitung

Upgrades von Forms 6i Client-Server-Applikationen auf die aktuelle Web-Architektur gibt es bereits seit über 10 Jahren. Technisch gesehen ist das Upgrade auf die aktuelle Version eine relativ einfache Übung - Vorgehensweisen und Stolpersteine sind bekannt. Spannend und immer wieder neu sind die Lösungsansätze, die gefunden werden müssen, um die bestehende Client-Funktionalität mit den neuen Möglichkeiten von Forms wieder zur Verfügung zu stellen. Der Vortrag gibt eine Übersicht über das Migrationskonzept der bestehenden Forms Applikation, sowie technische Erfahrungen, die im Verlauf des Projekts mit dem neuesten Forms Release gesammelt wurden.

Fakten zur Applikation

Die Applikation ist eine Individualentwicklung im Bereich der öffentlichen Verwaltung eines Schweizer Kantons. Die Entwicklung wurde Anfang der 90er Jahre basierend auf Forms Version 2 begonnen und bis heute entsprechend den fachlichen Anforderungen ausgebaut und aktualisiert. Die Applikation verfügt über zahllose Funktionen, um Geschäftsdaten und Stammdaten zu bearbeiten. Eingehende Geschäftsdokumente werden gescannt und als Bilder abgelegt, aber nicht elektronisch ausgewertet. Die Inhalte der Dokumente werden in definierten Geschäftsprozessen als strukturierte Geschäftsdaten abgelegt oder bestehenden Geschäftsdaten zugeordnet. Ausgehende Dokumente werden in der Regel mit Word-Vorlagen als Serienbrief erstellt. Die Applikation selektiert und exportiert dazu die benötigten Geschäftsdaten aus der Datenbank und stellt diese in Form von Dateien den Word-Vorlagen zur Verfügung. Die Verarbeitung und Aufbereitung der Daten erfolgt dann mittels Word-Makros. Die Erfassung weiterer Geschäftsdaten erfolgt über Excel-Dateien. Die Applikation überträgt strukturierte Geschäftsdaten in eine bestehende Excel-Vorlage und speichert diese im Dateisystem des Clients. Die Excel-Dateien werden manuell vervollständigt und anschließend über die Applikation wieder in das System eingelesen. Die Ablage aller Geschäftsdaten erfolgt zentral in einer Oracle Datenbank. Die Applikation wird auch auf Tablet PCs mit einer eigenen lokalen Datenbank betrieben. Dazu können Geschäftsdaten aus der zentralen Datenbank exportiert und in die lokale Datenbank importiert werden. Der Datenabgleich erfolgt wiederum durch Export und Import der lokalen Daten in die zentrale Datenbank.
Technisch gesehen besteht die Applikation aus ca. 330 Forms Modulen, mit einem zentralen Menü Modul. PL/SQL Libraries oder Object Libraries sind nicht vorhanden. Ausnahme ist eine PL/SQL Library mit verschiedenen Funktionen für die bestehenden Oracle Reports. Das Datenmodell besteht aus ca. 200 Tabellen und einigen Views. PL/SQL Code in der Datenbank ist ebenfalls kaum vorhanden. Die gesamte Geschäftslogik ist in den Forms Modulen mit ca. 230.000 Zeilen PL/SQL Code abgelegt. 100 Forms Module verwenden das HOST Built-in, um Befehle auf dem Client abzusetzen. Insgesamt wird sehr viel externe Funktionalität auf dem Client verwendet.

Das Projekt – Anforderungen und Aufgaben

Der eigentliche technische Treiber des Projektes ist nicht die Forms Applikation selbst, sondern die Datenbank. Die Applikation wird heute mit Forms Version 6i Client-Server und der Datenbank

Version Oracle 9i betrieben. Die Version der Datenbank soll auf die aktuelle Version 11g Release 2 angehoben werden. Überlegungen oder Tests, die bestehende Forms 6i Applikation mit der 11g Datenbank zu betreiben, wurden nicht angestellt. Eine Migration der Forms Applikation in Richtung moderne Technologien wie Java oder Microsoft wurde aus strategischen und kostentechnischen Gründen verworfen. Daraus entstand die Anforderung, die Forms Applikation auf eine aktuelle Version anzuheben, so dass die Applikation in den nächsten 3 bis 5 Jahren weiterhin betrieben und erweitert werden kann. Gleichzeitig wird auch eine Umstellung der Client Software und der Client Architektur geplant. Das Betriebssystem der Forms Clients wird von Windows XP 32-bit auf Windows 7 64-bit geändert, sowie Office 2003 durch Office 2010 ersetzt. Der spezielle Forms Client für diese Applikation soll auch als Citrix Client zur Verfügung stehen. Der Einsatz von geeigneten Tools für die Umstellung der Forms Module wurde ebenfalls aus Kostengründen verworfen. Im Zuge des Upgrades sollen häufig verwendete Komponenten und Funktionen identifiziert und als zentrale Basisfunktion aufgebaut werden. Ein Parallelbetrieb der alten und neuen Version wurde nicht geplant. Sämtliche Client Funktionen sollen weiterhin auf dem Client bleiben und nicht durch Server Funktionen ersetzt oder entsprechend umgebaut werden. Die Umstellung der Datenbank Version, des Client Betriebssystem und der Office Version gehören nicht zu den Aufgaben des Projektes. Allerdings benötigt die neue Forms Version Application Server. Zu den Projektaufgaben gehört der Aufbau von zwei Application Server Umgebungen auf einem Windows 2008 Server 64-bit Betriebssystem. Eine Umgebung steht für die Entwicklung und Test zur Verfügung, eine weitere Umgebung für den produktiven Betrieb. Eine Umstellung der Tablet PCs wurde verworfen. Die bestehenden Systeme haben nicht genügend Leistung, um die benötigte Oracle Software für die Datenbank und die Forms Umgebung zu betreiben. Außerdem wären für die lokalen Forms Umgebungen hohe zusätzliche Lizenzkosten entstanden.

Das Konzept

Die wesentliche Aufgabe im Projekt besteht darin, die Applikation mit allen Funktionen eins zu eins in der neuen Forms Version wieder zur Verfügung zu stellen. Dabei sind zusätzlich zum Upgrade der Forms Version die Umstellung der Datenbank Version, des Client Betriebssystem und der Office Version zu berücksichtigen. Die Vorgehensweise für das Upgrade ist bekannt und einfach. Die Forms Module werden einmalig mit dem Forms Migration Assistant auf die neue Version angehoben. Alle weiteren Anpassungen der Forms Module insbesondere für die Client Funktionen erfolgen manuell über den Forms Builder. Ein geeignetes Java Programm für die Suche von Begriffen in den Forms Modulen steht zur Verfügung. Mit diesem Suchprogramm werden die Forms Module identifiziert, bei denen Built-ins wie Text_IO oder HOST durch entsprechende WebUtil Befehle umgestellt werden müssen. Für die Zentralisierung von Basisfunktionen wird eine neue PL/SQL Library angelegt, die nahezu alle Client Funktionalitäten der Applikation modular in PL/SQL Packages kapselt. Die technische Umstellung wird begleitet und unterstützt durch die Fachabteilung. Die Fachabteilung testet frühzeitig die umgestellten Forms Module und protokolliert die Fehler.

CodeSearch

Für die Suche von Begriffen wie Text_IO oder HOST im PL/SQL Code von Forms Modulen wurde im Vorfeld des Projektes ein Java Programm basierend auf Forms JDAPI entwickelt. CodeSearch ist Command-Line gesteuert und ermöglicht die Eingabe von Suchbegriffen und Datei-Filtern.

```
java com.trivadis.forms.runtime.RuntimeManager -c=codesearch -
d=D:\Development\Forms\work -f=*.fmb -s=host
```

Gesucht wird dann in allen PL/SQL Einheiten der Module. Die Treffer mit Bezug zum Kontext werden in Logdateien ausgegeben.

```
Forms Migration Repository Version 2.0
------------------------------------------------------------------------
command.............................. : codesearch
```

```
directory.............................. : D:\PePDevelopment\Forms\work
filter................................. : *.fmb
logfile................................ : codesearch.log
------------------------------------------------------------------------------
starttime.............................. : 04.09.2012 08:40:29.995
files=327
------------------------------------------------------------------------------
Print the results
------------------------------------------------------------------------------
search.filename=D:\PePDevelopment\Forms\gsttg\exe\gesaf001.fmb
search.filename.hits=2
GESAF001:Blocks:GESA:Items:ITEM18:Triggers:WHEN-BUTTON-PRESSED host 48
host(l_befehl,1);
GESAF001:Blocks:GESA:Items:ITEM15:Triggers:WHEN-BUTTON-PRESSED host 21
host(l_befehl,1);
------------------------------------------------------------------------------
…
------------------------------------------------------------------------------
endtime................................ : 04.09.2012 08:40:50.042
elapsedtime............................ : 00:00:20.047
------------------------------------------------------------------------------
search.hits=206
```

Bei der Anzahl der Forms Module ist eine solche Funktion für ein Upgrade unbedingt erforderlich.

Parametersteuerung

Die Parametersteuerung der Applikation erfolgt über eine INI-Datei, die auf jedem Forms Client verfügbar ist. Zentrale Parameter werden im Startmodul der Applikation gelesen und in globalen Variablen gespeichert, damit andere Forms Module darauf zugreifen können. Da eine dezentrale Parametersteuerung mit individuellen Anpassungsmöglichkeiten für jeden Forms Client nicht mehr erforderlich ist, werden die Parameter der INI-Datei in einer neuen Tabelle der Datenbank abgelegt. Für den Zugriff wird ein PL/SQL Package entwickelt, mit dem ein Parameterwert anhand des Parameternamens eindeutig identifiziert werden kann. Für die zentrale Verwaltung der Parameter wird ein neues Forms Modul entwickelt. Durch diese Anpassung konnten die sequentiellen Leseroutinen mit Text_IO für die benötigten Parameter aus der INI-Datei in ca. 50% der Forms Module durch einfache Funktionsaufrufe ersetzt werden. Prüfungen, ob die Parameter vorhanden und Werte verfügbar sind, werden nicht mehr benötigt. Sie sind jetzt Bestandteil des Verwaltungsmoduls.

Text_IO und HOST Befehle ersetzen

Das durchgängige Ersetzen der Text_IO Befehle in 80 Forms Modulen durch Client_Text_IO hätte auch programmatisch erfolgen können. Die manuelle Korrektur erfordert mehr Zeit, gibt aber die Möglichkeit den Programmablauf mit SYNCHRONIZE Befehlen zu ergänzen und somit Fehler in der Verarbeitung zu vermeiden. Gleiches gilt für das Ersetzen der HOST Befehle in 100 Forms Modulen durch WEBUTIL_HOST. Dabei sind aber in der Regel die gleichen Forms Module betroffen, wie bei den Text_IO Korrekturen, so dass der zusätzliche Aufwand insgesamt vertretbar ist. Das manuelle Vorgehen ermöglicht die Ergänzung von SYNCHRONIZE Befehlen im Programmablauf und die Unterscheidung zwischen blockierenden und nicht blockierenden Aufrufen. Dadurch können weitere Fehlerquellen in der Verarbeitung vermieden werden. Um die Wartbarkeit, die Kontrolle und die Protokollierung der Client-Aufrufe zu verbessern, wird für alle Kommandos ein PL/SQL Package in der zentralen PL/SQL Library angelegt.

DDE ersetzen durch Client_OLE2

Da die Microsoft Office Anbindung weiterhin auf dem Forms Client erfolgen sollte, müssen die Funktionen aus dem DDE Package zum Schreiben und Lesen von Excel-Dateien komplett durch

Client_OLE2 Funktionen ersetzt werden. Ein zusätzliches Package in der zentralen PL/SQL Library stellt alle benötigten Funktionen zur Verfügung. Die Umstellung der PL/SQL Befehle ist einfach. Auch die Umstellung auf Excel 2010 bringt keine unerwarteten Probleme. Die Applikation erstellt mehrere Excel-Dateien in einer Schleife. Der Programablauf wird mehrfach korrigiert, um die nötige Stabilität zu erreichen. Beim Lesen der Excel-Dateien gibt es Fehler beim Auslesen und Konvertieren einer Zelle mit Datum. Die Konvertierung mit einer Oracle Datumsformatmaske ergibt immer wieder falsche Werte. Eine Lösung konnte durch eine eigene Konvertierungsroutine erreicht werden.

RUN_REPORT ersetzen durch RUN_REPORT_OBJECT

Die Umsetzung der Berichte innerhalb der Applikation erfolgt hauptsächlich mit PL/SQL-Einheiten in den Forms Modulen, die HTML-Ausgaben erzeugen. Andere Dokumente wie Briefe und Formulare werden über Word-Vorlagen und Serienbriefe abgerufen. Insgesamt sind in der Applikation zwei Reports vorhanden, die für den Ausdruck von Rechnungen und Zahlungsträgern erforderlich sind. Die RDF-basierten Berichte enthalten Parameter-Formulare, die als Forms Module neu implementiert werden. Für den Aufruf der Berichte aus den Forms Modulen wird RUN_REPORT_OBJECT verwendet. Der Abruf der erstellten PDF Dokumente erfolgt über das RWServlet des Reports Servers, die Anzeige mit WEB.SHOW_DOCUMENT. Für diese Funktionen wird ein eigenes PL/SQL Package entwickelt und mittels PL/SQL Library zentral zur Verfügung stellt.

Import, Export und SQL*Loader auf dem Client

Pragmatisch ist der Lösungsansatz in Bezug auf den Oracle Datenbank Client. Benötigt werden die Datenbank-Utilities Export, Import und SQL*Loader, sowie SQL*Plus. Auf den Forms Clients wird die Datenbank Client Software 11g Release 2 installiert, um den entsprechenden Client zur Datenbank zu verwenden. Da die Tablet PCs weiterhin mit Forms 6i und der Oracle 9i Datenbank arbeiten, muss auch eine Oracle 9i Datenbank Client installiert werden. Der Datenaustausch zwischen den lokalen Systemen der Tablet PCs und der zentralen Datenbank erfolgt über die Export und Import Utilities. Auf den Tablet PCs steht nur der 9i Client zur Verfügung. Die exportierten Daten der Tablet PCs können nur mit dem 9i Client erfolgreich in die 11g Datenbank importiert werden. Die Daten der 11g Datenbank dürfen nur mit dem 9i Client exportiert werden, damit der Import am Tablet PC erfolgreich ist. Der SQL*Loader aus dem 11g Client wird für das Laden von externen Daten verwendet.
Da die Konfiguration der benötigten Pfade auf dem Client Betriebssystem nicht eindeutig für beide Datenbank Clients gelöst werden konnte, wurden spezielle Windows Command Dateien erstellt, die die richtigen Umgebungsvariablen vor dem Aufruf setzen. Für die Forms Applikation wird ein neues PL/SQL Package mit Aufrufen für die Programme aus dem jeweiligen Datenbank Client entwickelt und über eine PL/SQL Library zentral zur Verfügung gestellt.

SQL*Plus auf dem Client

Zur Applikation gehört eine Sammlung von 165 SQL-Skripten, die mit SQL*Plus ausgeführt werden. Neben einfachen SQL-Auswertungen mit SELECT-Anweisungen gibt es auch Skripte zur Korrektur von Geschäftsdaten. Teilweise werden neue Datenbanktabellen temporär aufgebaut und wieder gelöscht. Der erste Lösungsansatz war einfach gedacht, die HOST Befehle in den Forms Modulen werden auf WEBUTIL_HOST Befehle umgestellt und SQL*Plus aus dem 11g Datenbank Client aufgerufen. Doch die einfache Umstellung ist nicht möglich. Das grafische Benutzerinterface für SQL*Plus unter Windows (sqlplusw.exe) gibt es in dem Datenbank Client 11g nicht mehr. Zitat aus den Release Notes der Datenbank 11g:

```
The SQL*Plus for Windows graphical user interface (GUI) was desupported in
SQL*Plus Release 11.1.
```

Die Command-Line Variante (sqlplus.exe) kann zwar über Forms aufgerufen werden, der SQL*Plus Prozess bleibt aber für den Anwender verborgen. Bei allen Skripten ohne Parametereingabe mit ACCEPT (ca. 80) eine mögliche Variante, bei allen anderen nicht. Bei vielen SQL-Skripten mit

Parametern handelt es sich um Benutzereingaben, die als Kriterien für die SELECT-Anweisungen dienen. Dafür wurde als Prototyp ein einfaches Forms Modul entwickelt, welches die Parametereingaben ermöglicht. Dazu sollten alle Skripte und Parameter in der Datenbank als Metadaten erfasst werden. Im Forms Modul wird das gewünschte Skript ausgewählt und die erforderlichen Parameter aus den Metadaten gelesen und abgefragt. Die Eingaben können dann beim Starten des Skriptes übergeben werden, so dass keine Benutzerinteraktion erforderlich ist. Andere SQL-Skripte beinhalten allerdings ACCEPT-Anweisungen, um die Skripte bei Bedarf abzubrechen oder das COMMIT-Verhalten zu beeinflussen. Ein solches Workflow-Verhalten ist vorab durch entsprechende Eingaben nicht abzubilden, so dass der gesamte Lösungsansatz aus Zeitgründen verworfen wurde. Schlussendlich wurde der gleiche pragmatische Lösungsansatz wie bei den Datenbank-Utilities Export, Import und SQL*Loader gewählt. Der 9i Datenbank Client verfügt noch über das grafische Benutzerinterface für SQL*Plus unter Windows und kann alle Skripte unverändert auch gegen eine 11g Datenbank ausführen. Langfristig ist das keine geeignete Lösung, da der 9i Datenbank Client irgendwann nicht mehr benötigt wird oder nicht mehr mit höheren Datenbankversionen funktioniert. Vernünftig wäre eine Ablösung durch PL/SQL Einheiten in der Datenbank.

Scanner Integration auf dem Client

Das Scannen der eingehenden Geschäftsdokumente gehört zu den wichtigsten Funktionen der Applikation. Die Dokumente werden zunächst in den Scanner gelegt. Die Ansteuerung der Scanner erfolgt über die Zusatzsoftware Twain2File. Das Programm wird von der Forms Applikation aufgerufen und startet den Scan-Vorgang. Die Dokumente werden als nummerierte Bilddateien in einem konfigurierten Verzeichnis abgelegt. Die Forms Applikation übernimmt wieder die Kontrolle, liest die Bilddateien in die Datenbank ein und erzeugt eine entsprechende Referenz zu den Geschäftsdaten.

Die Umstellung war schwierig und ist auch noch nicht ganz abgeschlossen. Die Treiber und Utilities für die eingesetzten Epson Scanner wurden auf Windows 7 64-bit umgestellt. Die eingesetzte Zusatzsoftware für die Steuerung der Twain Schnittstelle zum Scanner ist nicht für das neue Betriebssystem verfügbar und wurde durch IrfanView ersetzt, bietet aber nicht ganz die gleiche Funktionalität. WebUtil ermöglicht die Identifikation der erzeugten Bilddateien und das Einlesen in die Datenbank. Die Funktionalität ist verfügbar, aber die Kontrolle der externen Prozesse durch die Forms Applikation ist problematisch und die Performance ist deutlich schlechter.

Windows Scripting Host auf dem Client

Für direkte Interaktionen mit dem Windows Client wird Windows Scripting Host (WSCRIPT) in Kombination mit verschiedenen VB Skript Routinen eingesetzt. Die Funktionen rufen Client Programme wie Internet Explorer, Word und Excel auf oder ermöglichen Dateioperationen auf dem Client wie Kopieren, Verschieben, Löschen und das Setzen von Dateiattributen. Weitere VB Skripte ermöglichen die Kommunikation mit dem lokalen Mail Client zur Versendung von Mails sowie die Kommunikation mit einem Web Service zur Abfrage von GIS Daten.

Trotz der Umstellung des Betriebssystems, der Office Version und der Forms Version sind die Funktionen ohne Anpassungen verfügbar. Probleme entstehen hier immer wieder bei der Synchronisation zwischen der Forms Applikation und den Client Programmen. In verschiedenen Forms Modulen werden HTML-Dateien auf dem Client erzeugt und anschließend in einem weiteren Browser Fenster angezeigt. Der Internet Explorer wird dazu mit WSCRIPT gestartet und erhält als Übergabeparameter den lokalen Dateinamen, um die HTML-Datei anzuzeigen. Das funktioniert nicht immer. Erst die Ergänzung eines SYNCHRONIZE Befehls nach dem Schließen der Datei bringt die benötigte Stabilität.

```
declare
   l_filename       varchar2(400);
   l_buffer         varchar2(4000);
```

```
   l_file           client_text_io.file_type;
   l_cmd            varchar2(4000);
begin
   l_filename := 'C:\workdir\myfile.html';
   l_file := client_text_io.fopen(l_filename);
   l_buffer := '<html><body>Text</body></html>';
   client_text_io.put_line(l_buffer);
   client_text_io.fclose(l_file);
   synchronize;
   l_cmd := 'cmd /c wscript apputil.vbs ' || l_filename;
   webutil_host.non_blocking(l_cmd);
end;
```

Die Probleme entstehen hier, weil die Forms Applikation die Prozesse auf dem Client nur starten, aber nicht steuern oder kontrollieren kann. Eine direkte Kommunikation besteht nicht. Solche Problemzonen sind im Programmcode nur schwer zu identifizieren und die Fehler sind teilweise nicht reproduzierbar.

Anpassung Farben und Schriftarten mit Forms JDAPI

In der Schlussphase des Projektes wurden Farben und Schriftarten der Oberfläche angepasst, so dass die ursprünglichen DEFAULT Einstellungen wie grauer Hintergrund und weiße Felder mit schwarzer Schrift wieder zur Geltung kamen. Für die Applikation wurde dazu ein zentrales Forms Modul mit den benötigten Standardobjekten wie Window, Canvas, verschiedene Items und Visual Attributes angelegt. Die Objekte haben einige definierte Standardeigenschaften wie Höhe und Breite und verwenden die Visual Attributes für die Definition von Farben und Schriftarten. Eine Object Group dient als Container, um die visuellen Attribute einfach in jedes Forms Modul übertragen zu können. Alle Objekte aus diesem zentralen Forms Modul wurden in eine zentrale Object Library kopiert.
Für die Übertragung der Referenzen auf die bestehenden Forms Module wurde ein Java Programm basierend auf Forms JDAPI entwickelt. Das Programm analysiert Window, Canvas und Item Objekte der Forms Module und trägt die verfügbaren Referenzen (Subclassing) aus der Object Library ein. Zukünftig kann die gesamte Applikation dadurch Farben, Schriftarten und Standardeigenschaften aus einer zentralen Object Library referenzieren.

Forms Fehler

Nach dem Upgrade der gesamten Applikation mit dem Forms Migration Assistant werden die ersten Tests durchgeführt. Einzelne Forms Module werden aufgerufen und Daten selektiert. Dabei stellt sich folgender Fehler ein, der zunächst nicht nachvollziehbar ist.

```
FRM-93652: The runtime process has terminated abnormally.
Contact your system administrator.
```

Die Fehlermeldung gibt keinen direkten Hinweis auf die eigentliche Fehlerursache. Die Suche beim Oracle Support ergibt entsprechende Hinweise. Eine Reihe von möglichen Fehlerquellen und Lösungen sind in der Master Note 1297540.1 Known Causes of the FRM-93652 Error In Forms zusammengefasst. Die Fehlerursache ist eine Property Einstellung in einem Menü Modul. Bei allen Menü Modulen hat die Eigenschaft „Share Library with Form“ den Wert „No“. Einige Forms Module haben einen Trigger KEY-ENTQRY auf Forms Level, der die eingegebenen Query Kriterien vor der Ausführung (Execute_Query) überprüft. Wenn keine Kriterien eingegeben werden erfolgt eine Sicherheitsabfrage, ob das wirklich so gewünscht ist, da die Query entsprechend lange dauern kann. Die Abfrage verwendet das Built-in FIND_ALERT. Diese Kombination führt zum Absturz der Forms Applikation. Der Fehler tritt nicht mehr auf, wenn bei den Menü Modulen die Eigenschaft „Share Library with Form“ auf den Wert „Yes“ gesetzt wird.
Genaue Details zu diesem Fehler sind im Oracle Support Dokument 1096125.1 Forms Crash With Error FRM-93652 After 11g Upgrade zu finden.

Die gleiche Fehlermeldung tauchte in einem anderen Zusammenhang wieder auf. Text_IO wird in allen Forms Modulen durch Client_Text_IO ersetzt. Die funktionalen Tests dieser Module, die in der Regel HTML-Dateien auf dem Client erzeugen sind erfolgreich und erzeugen keine Fehler. Bei weiteren Tests mit größeren Datenmengen stürzt die Forms Applikation mit dieser Fehlermeldung ab. Die Ursache liegt hier in der Verwendung von Client_Text_IO in einer Schleife, in der mehrere 10.000 Datensätze geschrieben werden. Der Fehler konnte durch die Ergänzung eines SYNCHRONIZE Befehls innerhalb der Schleife nach jeweils 1.000 Datensätzen behoben werden.
Details zu diesem Fehler und der Lösung sind im Oracle Support Dokument 1231693.1 Forms with WEBUTIL's CLIENT_TEXT_IO in a Loop Thousands of Times, Hangs OR Crashes (FRM-93652) beschrieben.

Forms Server Fehler
Etwas kurios ist der folgende Fehler beim Starten der Forms Applikation, der im Projekt dann auftrat, wenn zuvor ein Aufruf eines Oracle Reports erfolgte.

```
Failure of Server APACHE Bridge
```

Kurios insofern, als es für diesen Fehler ein Patch gibt, dass in der installierten Version 11.1.2.0.0 laut OPatch bereits enthalten ist. Erst das erneute Einspielen des Patches 12632886 bereinigt den Fehler endgültig. Genaue Details zum Fehler sind zu finden im Oracle Support Dokument 1380762.1 Forms / Reports 11g Intermittent FRM-92103 Or "Failure of Server APACHE Bridge" - OHS Log Shows "apr_socket_connect call failed".

Schließen des Browser Fensters
Relativ schnell entsteht die zusätzliche Anforderung, dass Browser Fenster beim Beenden der Forms Applikation automatisch zu schließen. Aus anderen Projekten ist bekannt, dass eine Lösung mit JavaScript möglich ist. Eine gute Anleitung ist im Oracle Forms Community Blog zu finden. Im ersten Schritt wird eine HTML-Datei close.html erstellt. Für Mozilla und andere Browser sieht der HTML-Code so aus:

```
<html>
<body onload="closeit()">
<script>
function closeit() {
   window.close();
}
</script>
</body>
</html>
```

Für den Internet Explorer sieht der HTML-Code so aus:

```
<html>
<body onload="closeit()">
<script>
function closeit() {
   win = top;
   win.opener = top;
   win.close ();
}
</script>
</body>
</html>
```

Die HTML-Datei muss im zweiten Schritt in einem Verzeichnis auf dem Application Server abgelegt werden, dass über den HTTP Server erreichbar ist. Am einfachsten ist die Verwendung des

DocumentRoot Verzeichnisses (htdocs). Jedes andere verfügbare virtuelle Verzeichnis des HTTP Servers kann ebenso benutzt werden.
Im dritten Schritt wird die Forms Applikation erweitert. Im zentralen Startmodul der Applikation wird ein Post-Form-Trigger ergänzt. Der Trigger enthält folgenden PL/SQL-Code:

```
begin
   WEB.SHOW_DOCUMENT('http://machinename.domain/close.html','_self');
end;
```

Beim Beenden der Forms Applikation wird die HTML-Datei aufgerufen, die per JavaScript das aufrufende Browser Fenster schließt.
Der Beitrag und das Beispiel sind im Internet unter folgender Adresse zu finden:
http://oracleformsinfo.wordpress.com/2011/12/19/how-can-i-close-the-parent-browser-window-while-running-forms-on-the-web/

Login mit LOCAL Parameter
Wenn der LOCAL Parameter gesetzt ist, muss der Datenbankname beim Login nicht eingegeben werden. Für das Forms Login wird der LOCAL Parameter in der env-Datei der Applikation gesetzt.

```
LOCAL=<dbname>
```

Der Vorteil beim Login hat allerdings seine Nachteile. Der Aufruf des Forms Built-ins

```
Get_Application_Property (CONNECT_STRING)
```

gibt keinen Wert zurück. Der LOCAL Parameter muss dann konsequent in allen anderen Umgebungen, die die Login Informationen benötigen, ebenfalls gesetzt werden. Dazu gehörten in diesem Projekt der Forms Client, wegen der SQL*Plus Aufrufe und der Reports Server, wegen der RUN_REPORT_OBJECT Aufrufe. Alternativ kann der Wert des LOCAL Parameters mit dem Forms Built-in ToolEnv.GetVar abgefragt werden.

Erfahrungen und Fazit
Die neueste Forms Version 11g Release 2 64-bit ist stabil und zuverlässig. Die Installation ist im Vergleich zu den Installationen von 11g Release 1 einfacher und schneller. Die vorhandenen Fehler der Software konnten identifiziert und behoben werden. Das Konzept für das Upgrade und die Vorgehensweise im Projekt haben sich bewährt, so dass die bekannten Probleme und auch die neuen technische Herausforderungen gelöst werden konnten.
Die Applikation enthält viele Funktionen, die auf den Forms Client zugreifen und dort ausgeführt werden. Ein Umbau dieser Client-Funktionen wäre im Laufe des Upgrades möglich und teilweise auch nötig gewesen. Darauf wird zunächst verzichtet, aber für die Zukunft eingeplant. Durch das Projekt kann die Applikation in einem überschaubaren Zeitrahmen und mit einem relativ kleinen Budget wieder aktualisiert werden und hat durch das Upgrade eine vernünftige Perspektive für die nächsten 3 bis 5 Jahre.

Kontaktadresse
Perry Pakull
Trivadis AG
Europa-Strasse 5
CH-8152 Glattbrugg

Telefon:	+41-44-808 70 20
Fax:	+41-44-808 70 21
E-Mail	perry.pakull@trivadis.com
Internet:	www.trivadis.com