Alle Wege führen zu Forms 12*c*

Frank Hoffmann, Cologne Data GmbH

Am 23.Oktober 2015 hat Michael Ferrante, Produktchef von Oracle Forms, über den Twitter-Account „@OracleFormsPM" die neue Version 12*c* von Forms ausgerufen. Damit hatte die lange Zeit des Wartens ein Ende und viele Gerüchte konnten mit Fakten beseitigt werden.

Oracle Forms hat eine Zukunft – bis mindestens zum Jahr 2023; trotz nebulöser Gerüchte, die etwas anderes behauptet haben, ist Reports12*c* auch an den Start gegangen und lebt eine ganze Applikations-Generation weiter. Wer bis Anfang des Jahres fleißig Änderungsvorschläge an Oracle für die neue Forms-Version geschickt hatte, konnte nun die eine oder andere Anregung in der Umsetzung wiederfinden. Dennoch ist die neue Version Forms 12*c* ein konservatives Funktions-Upgrade zu der Vorgängerversion Forms 11*g*R2 geworden. Wichtige Bibliotheken („webutil.pll") und Funktionen wurden „1:1" übertragen, Forms-12*c*-Dokumente aus bestehenden 11*g*R2 Dokumenten kopiert und Reports hat sich seit der Version 10*g* funktional nicht mehr geändert.

Auf der diesjährigen DOAG Konferenz hat Michael Ferrante die neuen Features von Forms 12*c* vorgestellt – eine gute Übersicht über die neuen Features. Die wichtigen Änderungen, die dieser Artikel beleuchtet, wurden „unter der Haube" durchgeführt. Erfahrungen aus mehrjähriger Migrationserfahrung sollen auch hier einfließen sowie schlagkräftige Argumente für Entscheider, warum ein Umstieg auf Forms 12*c* sehr sinnvoll ist.

Aus der Geschichte

Viele Oracle-Kunden haben sich Mitte der 1990er-Jahre für Forms entschieden, um damit ihre Datenbank-Applikationen zu entwickeln. Diese Wahl wurde zu jener Zeit auch von Oracle Consulting empfohlen und gefördert. Ein aus zwei GUI-Vorgängerversionen (4 und 5) gereiftes Release 6*i*R2 konnte direkt in der Produktion eingesetzt werden. Diese Begeisterungswelle gibt es leider nicht mehr, denn heute spricht man auf dem Markt mehr über Apex, ADF und reine Java-Anwendungen. Es gibt sicherlich auch Projekte, die sich damit gut realisieren lassen und die mit dynamischem HTML und JavaScript Eindruck machen.

Nicht wenige alte, aber nicht gealterte Entwicklungschefs behaupten bis heute, und der Autor zählt sich dazu, dass es für Projekte mit komplexen Datenbank-Modellen und Workflows kein besseres und kein produktiveres Tool gibt als Oracle Forms. Warum hat

	Forms 6*i*R2	Forms 11*g*R2	Forms 12*c*
Erscheinungsdatum	Sommer 2000	Januar 2011	Oktober 2015
Support (erweitert)	2005 (2008)	2016 (2018)	2020 (2023)
JDK 32-Bit	JDK 1.1	JDK 1.6	
JDK 64-Bit	-	JDK 1.7	JDK 1.8
Client-Umgebung	Installation Runtimeversion	JRE 1.7.40+ JRE 1.8.51+	JRE 1.7.55+ JRE 1.8.51+
PL/SQL-Version	8.0.6.3	11.1.0.7	11.2.0.3
Datenbank	Läuft „unsupported" auf 8..11*g* und 12*c* mit sqlnet Anpassung	10.2, 11*g*R1 R2, 12*c* R1	11.2.0.4, 12.1.0.2 + Datenbank Repository für jede Installation (DEV0..DEV99)
Zertifizierte Browser	-	Windows IE 8,9,10,11 FF24+ MAC IOS Safari 7+, 8+	Windows IE 11 FF 31+ MAC IOS (*) Safari 8+
SQLNET	8	11	11
Forms/Reports Client-Verbindung	via SQLNET 8 (Client/Server)	http/https aus Browser mit Plug-in JRE 1.7,1.8	http/https via JRE 1.8 oder Webstart (JNLP) oder JRE Stand-alone (dann ohne JavaScript und SSO)
Barrierefreiheit	Nein	Java Access Bridge mit JAWS12+	Java Access Bridge mit JAWS12+

Tabelle 1 (*) Ein Testaufruf von Forms 12c mit einem aktuellen MacBook hat gut geklappt. Nach Installation von JRE 1.8.66 für Mac wird das Applet gestartet. Der Java-Prozess der Forms-Anwendung wurde im Mac sogar zur Laufzeit in der Taskleiste angezeigt und wenige Sekunden nach Schließen des Browsers beendet.

Oracle das in den letzten Jahren nicht mehr so propagiert wie in den 1990er-Jahren? Warum wird reinen Java- und HTML-basierten Lösungen in der Regel mehr Priorität eingeräumt? Warum gibt es auf der DOAG Konferenz nur drei Forms-Vorträge – keinen einzigen davon von Oracle Deutschland? Niemand weiß das genau – aber es muss ja in den nächsten Jahren nicht so bleiben.

Was an Forms begeistert

Oracle Forms ist ein hochproduktives Tool zur Masken-Generierung. Wer einmal die Entwicklungslogik dieses Tools verstanden hat, kann mit PL/SQL und einigen Entwicklungsregeln Beträchtliches leisten. Die Logik sowie der Basis-Funktionsumfang mit Triggern, Prozeduren und Parametern sind relativ leicht zu lernen. Wer strukturiert programmiert, Logik in der Datenbank hält und beim Aufbau der Applikation Regeln befolgt (Styleguide, Notation, Inline-Dokumentation, strukturierte Entwicklung etc.), kann Software mit hoher Nachhaltigkeit erzeugen. Warum nicht heute mal ein neues Projekt mit Oracle Forms 12 beginnen?

Tabelle 1 veranschaulicht die wichtigsten Entwicklungen. Die Angaben dazu stammen aus den Zertifizierungs-Tabellen und Dokumenten von Oracle mit Stand am 9. Dezember 2015.

Forms 12*c* setzt auf Datenbanken und Browsern einer neueren Generation auf. Die Java-Prozesse laufen stabiler als in der Vorgängerversion. Für den Autor ist das Highlight der neuen Version der Aufruf ohne Browser („stand-alone").

Die Forms-12c-Server-Architektur

Für den Demoserver wurde Forms/Reports 12*c* auf einem Windows 2012 R2 Server installiert. Die HTML-Seiten sollten durch das lizenzfreie „mod_owa" dynamisch angezeigt werden. Als Datenbank wurde eine Oracle 12*c* Standard Edition gewählt und für den Testaufruf über SSL ein SSL-Zertifikat installiert. Bei der Auswahl des Zertifikatsherstellers sollte zuvor überprüft werden, ob der Anbieter auch in der Oracle-Java-CA-Liste auftaucht, denn sonst melden Browser wie Firefox und Safari die Adresse als nicht vertrauenswürdig (auch wenn keine selbst signierten JAR-Files installiert sind). Ein Apache-2.4-Reverse-Proxy routet in der Konfiguration die Verbindungen an Forms weiter. Reports wird in der Installation über „blobdestination" im Hintergrund betrieben und generiert PDF-Files in die Datenbank *(siehe Abbildung 1 und Listing 1)*.

Eine erhöhte Sicherheit könnte noch mit einer vorgeschalteten Maske erreicht werden. Diese kann eine Vielzahl von Sicherheitsprüfungen durchführen und über Global-Links die Passwörter der Ziel-Da-

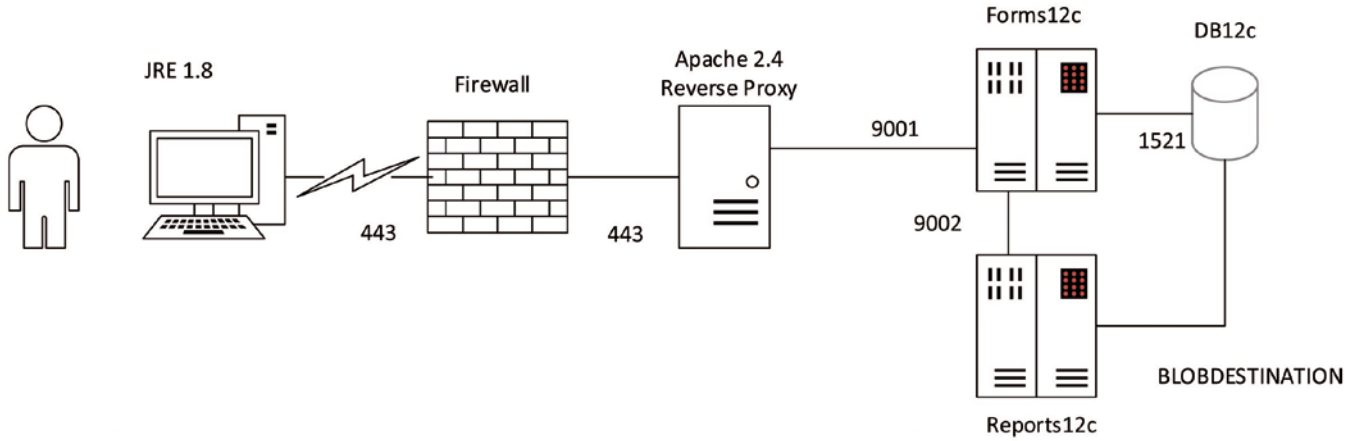

Abbildung 1: Forms-12c-Architektur mit einer verschlüsselten https-Verbindung

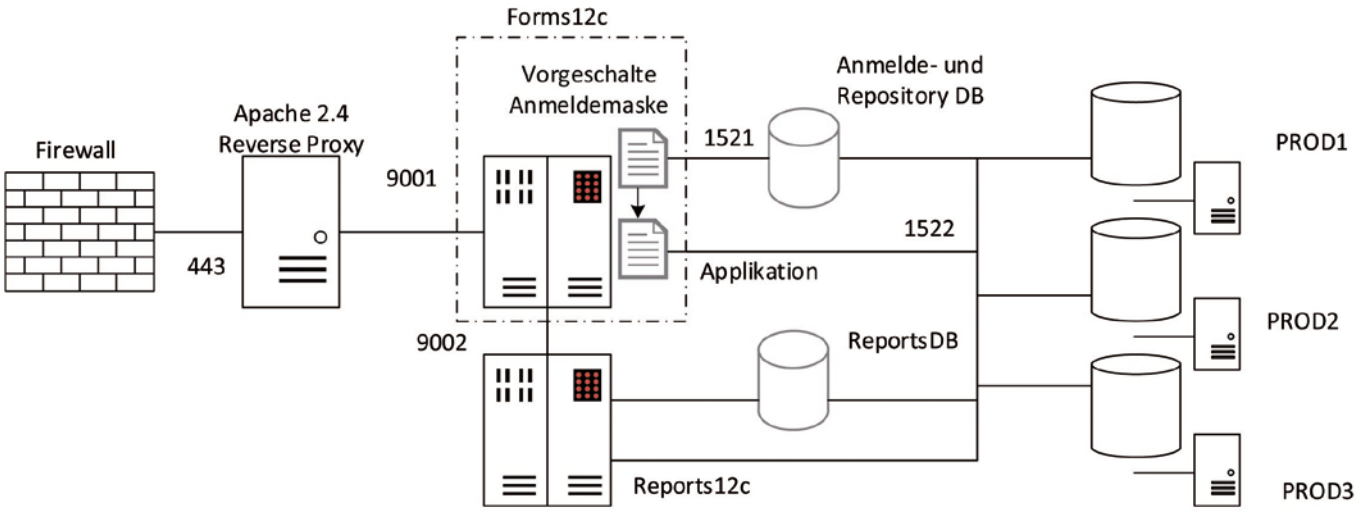

Abbildung 2: Erhöhte Sicherheit mit vorgeschalteter Anmeldemaske

tenbank prüfen, bevor eine Forms-Session gegen eine Produktions-Datenbank gestartet wird *(siehe Abbildungen 2 und 3)*.

Anmelde-Datenbank mit Anmeldemaske

Die Anmeldemaske verbindet sich automatisch mit der Anmelde-Datenbank beim Start der Applikation. So kann ein ungültiger Login schon protokolliert werden, bevor eine Anmeldung an die Produktions-Datenbank versucht wurde. Bei der Eingabe des Passworts prüft die Anmelde-Datenbank über einen Global-Link zur Ziel-Datenbank (Alias „PROTEST“ verbindet beispielsweise mit „PROD1“) das Passwort. Wenn es stimmt, wird mit „NEW“-Form eine Verbindung zu „PROD1“ über Port 1522 aufgebaut. Wenn es nicht stimmt, wird der Anwender abgewiesen.

Die Anmelde-Datenbank kann auch als zentraler Speicherort für das neue Forms 12*c*-Repository genutzt werden. Darüber hinaus ist auch eine zweite zentrale Datenbank zur Speicherung aller asynchronen Reports denkbar. Vorteil dabei ist, dass die Produktions-Datenbanken nicht unnötig mit Konfigurations-Schemata von Forms 12*c* und großen BLOB-Spalten von Reports belastet werden und die Online-Sicherungen überfrachten. Sicherheits-Anforderungen können abgedeckt werden durch:

- Abwehr von Denial-of-Service-Attacken – die Maske wird bei Massenaufruf von einer DOS-IP sofort beendet und eine Warnmeldung wird als E-Mail versendet
- Sicherheits-Verzögerungen bei mehrfachen Anmeldeversuchen
- Zulassung von maximal einem Anmeldefenster pro Client-ID

```
<VirtualHost colognedata.com:443>
   ProxyPreserveHost On
   ProxyRequests Off
   ServerName colognedata.com
   ProxyPass         /forms/ http://127.0.0.1:9001/forms/
   ProxyPassReverse  /forms/ http://127.0.0.1:9001/forms/
</VirtualHost>
```

Listing 1: Reverse-Proxy-Lösung zum Aufruf von Forms

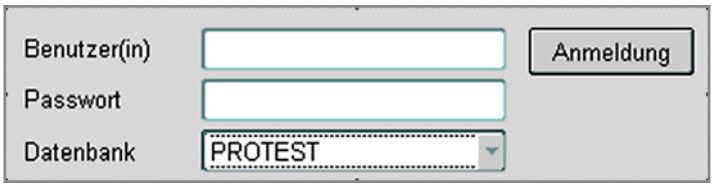

Abbildung 3: Login

- Verbote von Anmeldungen einer Kennung von verschiedenen Rechnern
- Abweisung kritischer Java- und Browser-Versionen
- Sicherheitshinweise per E-Mail bei Anmeldungen zu ungewöhnlichen Zeiten

Einstieg in die Welt von Forms 12c im Selbststudium

Ein guter Einstiegspunkt für die Beschäftigung mit Forms 12*c* ist neben dem Vortrag von Michael Ferrante auf der DOAG 2015 Konferenz die Oracle-Seite *„http://docs.oracle.com/middleware/1221/formsandreports/index.html" (siehe Abbildung 4)*.

Vor Beginn der Installation und den ersten Tests empfiehlt sich das Studium einiger weiterer Dokumente zu Systemvoraussetzungen *(siehe „http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-requirements-100147.html")*, Produkt-Zertifizierungen *(siehe „http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html")* sowie Forms Community und Support für 12*c*-Bugs *(siehe „https://community.oracle.com/community/development_tools/forms/content")*.

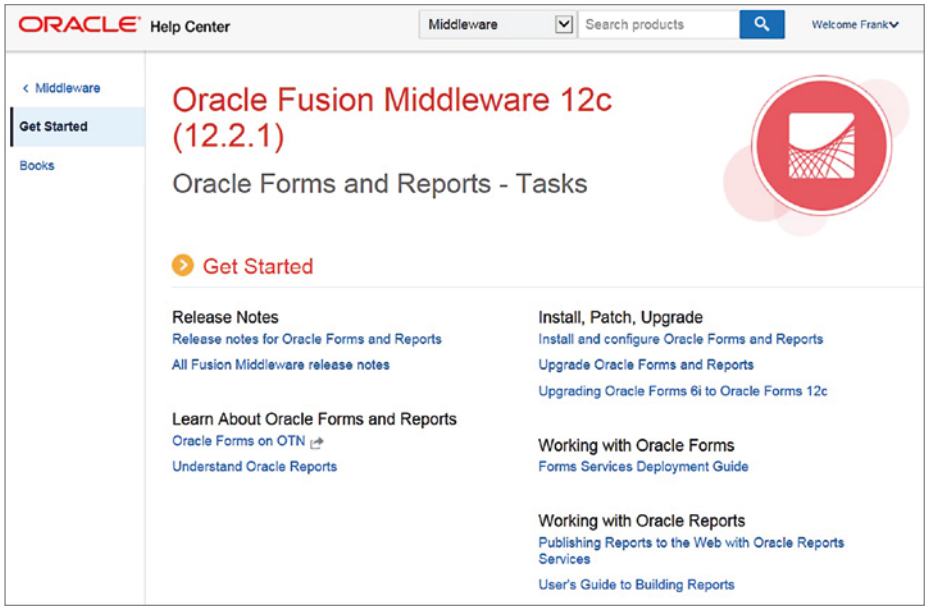

Abbildung 4: Oracle-Startseite

```
Default.env:
#FORMS_MODULE_PATH=%FORMS_PATH%
```

Listing 2: Anpassung der „default.env"

```
c:
cd C:\oracle\java
java -jar frmsal.jar -url "http://forms12c.com/forms/frmservlet?config=s
tandaloneapp" -t 30000
```

Listing 3: Stand-alone-JavaApplet-Start

Konflikt unter Forms 12c

Wer seine Masken und Reports in einem Verzeichnis halten und diese in dem Konfigurationsfile „formsweb.cfg" einrichten will, kann dafür die Applet-Variable „WorkingDirectory" verwenden. So wird ein gemeinsames Runtime-Verzeichnis für eine Konfiguration geschaffen. Unter Forms 12*c* muss dann aber im „default.env" die Variable „FORMS_MODUL_PATH" auskommentiert werden *(siehe Listing 2)*.

Neue Ereignis-Typen unter Forms 12c

Forms 12*c* kann mit der Außenwelt nun auch über neue System-Ereignisse kommunizieren. Neue Inaktivitäts-Timer können in Forms 12*c* definiert und durch Trigger abgefragt werden – das ging vorher nur mit Java Beans. Dazu ein Beispiel: Bei Maskenaufruf setzen wir die neuen Inaktivitäts-Timer mit „SET_APPLICATION_PROPERTY(CLIENT_IDLE_TIME,30);" auf 30 Sekunden. Dann definieren wir ein Ereignis mit dem Typ „System Client-Idle" und legen einen „WHEN-EVENT-RAISED"-Trigger mit dem Befehl „message(‚ClientIdle Trigger löst aus , || :system.last_event);" an. Das Ergebnis: Bei dreißig Sekunden Client-Inaktivität löst der Trigger aus.

Forms 12c als Stand-alone-JavaApplet komplett ohne Browser starten

Dieses Feature ist sehr nützlich und einfach zu implementieren. SSO und JavaScript fallen dann aber als Features weg. Zuerst muss dafür das „.jar"-File „frmsal.jar" auf dem Client installiert werden, beispielsweise vom Demo-Server des Autors *(siehe „http://forms12c.com/forms/html/fsal.htm")*. Dann kann man ein Batchfile *(siehe Listing 3)* lokal zum Start von Forms anlegen. Hier steht „frmsal.jar" im Verzeichnis „C:\oracle\java". Auf dem DOS Fenster erscheint dann ein Aufrufcode *(siehe Listing 4)*. Nach Schließen der Applikation wird das DOS-Fenster automatisch geschlossen.

Latenz-Messungen und Netzwerk-Analysen unter Forms 12c

Latenz-Messungen kann man weiterhin wie in Forms 11*g*R2 durch Hinzufügung von Umgebungsvariablen erreichen *(siehe Listing 5)*. Die Parameter müssen auch in dem jeweiligen „jpi"-Dokument bekannt gemacht werden.

Über die Seite *„http://forms12c.de/demo"* ist ein Aufruf der Beispiele jeweils in http oder https möglich. Auf der Seite sind auch die aufgeführten Links und Dokumente abgelegt sowie eine Kopie des Vortrags von Michael Ferrante auf der DOAG Konferenz zu Forms 12*c*. Der Latenz-Test wird durch Doppelklick auf die Anzeige der Roundtrips in dem entsprechenden Beispiel gestartet.

Grafiken aus einem Verzeichnis ohne „.jar"-File einlesen

Grafiken lassen sich aus 11*g*R2 oder 12*c* zur Laufzeit einlesen. Das wurde erfolgreich unter Forms 12*c* mit der Browser- und Standalone-Version getestet *(siehe Listing 6)*. Dazu müssen zwei Einträge gemacht und der WebLogic Server einmal neu gestartet werden – auch bei jeder zukünftigen Änderung im Grafik-Verzeichnis ist wegen der Cache-Einträge ein Neustart notwendig. Darüber hinaus muss ein neues „icons"-Verzeichnis unter „webutil" mit den entsprechenden GIF-Files angelegt sein.

Wechsel von älteren Forms-Versionen

Wer zurzeit eine aktuelle Oracle-Forms-Version betreibt, kennt das Problem mit Browsern und Java-Plug-ins. Es müssen Schutz-Mechanismen (JavaScript) eingebaut sein, damit nicht unbeabsichtigt „Forms-Browser"-Fenster geschlossen werden. Browser wie Firefox und Internet Explorer haben ihre Besonderheiten. Wie auch schon in der Tabelle sichtbar, sind bei jeder Version bestimmte Browser zertifiziert und andere nicht. Schwierig ist bei Browser-Abstürzen oder Verbindungs-Abbrüchen auch die Bereinigung der lokalen Client-Java-Sessions („Zombie"-Prozesse). Diese können das System instabil machen. Oft hilft hier für den Anwender nur eine Abmeldung vom Client-System. Einige dieser 11*g*-Bugs sind mit 12*c* gefixt worden. Die Migration beschränkt sich auf ein Batch-Recompile aller Module. Die Vorteile von Forms 12*c* gegenüber Forms 11*g*R2 sind:

- *Weg von Forms-Aufrufen über Browser-Plug-ins*
  Mit der neuen „Stand-alone"-Lösung lässt sich unter Forms 12*c* die Applikation ganz ohne Browser unter einer Java-Realtime-Umgebung (JRE) betreiben. Leider fallen hier aber auch Optionen weg. SSO und JavaScript werden dann nicht mehr unterstützt.

- *Support bis zum Jahr 2020*
  Ende des Jahres 2016 läuft der Support für 11*g*R2 aus und der kostenpflichtige „Extended Support" beginnt. Weiterhin hat Mozilla für Ende 2016 angekündigt, keine Java-Plug-ins mehr im Firefox zu erlauben. Wichtig ist auch zu prüfen, ob der Patch für die „.jar"-Files in 11*g*R2 eingespielt wurde. Die Zertifikate laufen sonst Ende Januar 2016 aus.

- *Optimiertes JVM-Speicher-Management von Forms und Reports*
  Ein Hauptproblem der Version 11*g*R2 ist das JVM-Speicher-Management. Hier verspricht die neue Version Bugfixes und einige Optimierungen. Das Record-Management eines Forms-Blocks wurde auch mit dieser Version in den Speicher verlegt. Der Report-Server muss bei der optimierten JVM-Speicher-Methode nicht mehr regelmäßig neu gestartet werden.

- *Integration BI Publisher*
  Neue Produkte wie der BI Publisher lassen sich ähnlich wie Reports integrieren.

- *WebUtil ohne OLE2 installieren*
  Wer auf OLE2 verzichten kann, bekommt mit Forms 12*c* die Chance einer Installation ganz ohne selbst signierte „.jar"-Files („webutil_no_ole"). Das ergänzende „.jar"-File für die Office-Automation, das bisher selber signiert werden musste („jacob.jar"), ist dann nicht mehr notwendig.

- *Neue System-Ereignistypen für Inaktivitätskontrolle*
  Neue Forms-12*c*-Ereignistypen können die Funktion des „timeout.jar"-Files ablösen. Das spart Entwicklungszeit und eine eigene Signierungs-Baustelle mit kostspieligem Code-Signing-Zertifikat.

Technische Argumente für einen Wechsel von 6i auf 12c

In der 6*i*-Umgebung kann nur mit PL/SQL in der Version 8 programmiert werden. Das ist der Stand des Jahres 2000. Neuere PL/SQL-Funktionen sind nicht unterstützt. Das gilt auch für den C-Compiler. Alter C-Code für Schnittstellen kann nur mit sehr, sehr alten Compilern gepflegt werden. Das mit Forms 6*i* verteilte JDK 1.1 ist ein Sicherheitsrisiko und bietet auch nur den Funktionsstand aus dem Jahr 2000. Die Anbindung an die Datenbank mit „NET8" ist veraltet. Eine Anbindung an eine Datenbank-Version 12 verlangt eine Anpassung der „sqlnet.ora". Die Argumente für die endgültige Verabschiedung von Forms 6*i* sind:

```
Archivdateien werden in das Verzeichnis C:\Users\COLOGN~1\AppData\Local\Temp\frmsal\12.2.1.0 heruntergeladen
Forms-Session-ID: WLS_FORMS.formsapp.51
Proxyhost: null, Proxyport: 0.
Native HTTP-Implementierung für Verbindung verwendet.
Verbindungsmodus: HTTP.
Version von Forms-Applet: 12.2.1.0
```

Listing 4

- *Der Betrieb von Forms 6i ist und bleibt ein Sicherheitsrisiko*
  Der Betrieb von Forms 6*i* im Intranet verlangt, wie auch der Betrieb über das Internet, einen geöffneten Datenbank-Port. Damit ist die Datenbank verwundbar. Einbrüche und Missbräuche der Accounts können über das Netz direkt gegen die Datenbank ausgeführt werden. Die geforderte SQLNET Version 8 bietet wenige Sicherheits-Funktionen und ist auf dem Stand des Jahres 2000. Passwörter gehen genau wie SQL-Code in der Regel unverschlüsselt über die Leitung. Alexander Kornbrust, ein Experte auf dem Gebiet der Datenbank-Sicherheit, hat schon früh mit vielen Beispielen darauf hingewiesen, wie groß das Sicherheits-Risiko eines offenen Datenbank-Ports ist – im Intranet schlecht, im Internet ganz übel. Der gesamte Datenbank-Verkehr läuft über die Leitungen und kann prima mitgelesen werden.

- *Eine lokale Datenbank-Installation wie in Forms 6i verrät zu viel*
  Forms 6*i* erfordert eine Runtime-Installation mit NET8. Wer die Datenbank-Verbindung kennt, kann mit 3rd-Party-Tools spielen, um eine unerwünschte Datenbank-Verbindung herzustellen. Ein Bösewicht kann auch leicht mit der Host-Adresse einen DOS-Angriff starten. Er kennt immer den Port und die Host-Adresse der Datenbank, an der er sich anmeldet – das ist aufwändig abzuwehren.

- *Passwörter können nicht im „Mixed-Case" betrieben werden*
  Dadurch, dass Oracle die Passwörter an Reports im „Upper Case" überträgt, fallen Passwörter mit Groß- und Kleinschreibung weg. Damit leidet die Sicherheit einmal mehr. Die Datenbank muss daher bei den Passwörtern auf Forms 6*i* Rücksicht nehmen und schränkt die Sicherheit bei der Passwort-Vergabe ein.

- *Mögliche Netzwerk-Belastungen mit 6i-Modulen*
  Es gibt Reports für den Massendruck oder lange Listen, die ein paar tausend Mal auf die Datenbank zugreifen. Zeile für Zeile werden dabei stundenlang die Datenbank und das Netz beschäftigt. In manchen Masken werden CLOB- oder BLOB-Spalten von Forms zeilenweise auf den Client übertragen. Hier ist mit dem Umstieg auf Forms 11*g*/12*c* eine deutliche Optimierung möglich. Erstens sind die Netzwege zwischen Datenbank und Report-Server kurz und gesichert, zweitens können die Reports asynchron im Hintergrund auf dem Applikationsserver ihre Arbeit verrichten und legen den Client nicht lahm. Auch Forms-Module mit viel Datenbank-Traffic laufen auf den neuen Forms-Versionen performanter.

```
Formsweb.cfg:
latencyCheck=true    # Latenzmessung von 1,4kb und 64 kb Paketen
networkStats=true# Anzeige Roundtrips u. Übertragungsvolumen
```

Listing 5: Latenz-Messungen unter Forms 12c

```
Formsweb.cfg: imageBase=documentBase
Registry.dat: default.icons.iconpath=webutil/icons/
```

Listing 6: Grafiken aus dem „webutil"-Verzeichnis einlesen

- *Sicherheitsrisiko bei der Verteilung der Module*
  Die Forms-Module zum Betrieb einer Applikation können beliebig getauscht werden. Damit ist das Risiko des Missbrauchs nicht zu unterschätzen. Die Authentizität der Anwendung ist nicht zu gewährleisten. Hier können einfach der Aufruf-Pfad der Module geändert und manipulierte Module eingeschleust werden.

Migration von 6i auf 12c

Einen Überblick über mögliche Änderungen und Anpassungen findet man in dem Upgrade Guide „Upgrading von Forms 6*i* nach 12*c*" *(siehe „http://docs.oracle.com/middleware/1221/formsandreports/upgrade-forms/index.html")*. Der Guide hat sich im Vergleich zur 11*g*R2-Version kaum geändert. Es gilt eigentlich daher fast alles, was auch schon in früheren Artikeln zu 11*g*R2 gesagt wurde. Prophetisch hat der Autor im Januar 2012 seinen DOAG-Vortrag daher auch „Erfahrungen bei der Migration von Forms/Reports 6*i* nach Forms/Reports 11.1.2 und 12" genannt und alles dort Gesagte ist eigentlich noch gültig. Nur das „c" hatte er nicht erraten.

In jedem Projekt sollte zu Anfang der alte 6*i*-Code analysiert werden, entweder durch Batch-Export der Module in XML oder Text oder durch schlaue API-Tools. Kritische Suchwörter wie „HOST", „OLE2", „FFI", „RUN_PRODUCT" oder „TEXT_IO" helfen, Masken aufzuspüren, die Probleme bereiten können. Grundsätzlich lassen sich alle Masken mit solchen Aufrufen automatisiert migrieren – der Aufruf unter Forms 11*g*/12*c* muss allerdings nach der Migration gründlich getestet werden.

WebUtil-Funktionen brauchen immer einen sichtbaren „Canvas", um funktionieren zu können – eventuell benötigen einige Menü-Aufrufe Hilfsmasken mit „webutil"-Canvas. In Forms 6*i* konnten viele Varianten programmiert werden, die unter 11*g*R2 nicht mehr oder anders laufen. Unter Unix müssen eventuell Windows-Schriftarten nachinstalliert werden – sonst können Reports unter Unix schon mal in fremdartigen Zeichen erscheinen.

Wird OLE2 genutzt, muss ein signiertes „jacob.jar"-File eingebunden sein. Es ist sinnvoll, ein Konzept zu erarbeiten, um neue 12*c*-Logik während der Migration automatisiert und standardisiert einzubinden. Trigger, Bibliotheken und Kommentare sollten in allen Modulen einheitlich eingepflegt werden. Etwas Grafik-Gestaltung wird nötig sein: Alle Icons müssen mit transparentem Hintergrund in „.gif"-Files konvertiert werden. In diesem Zuge könnte man auch Logos anpassen und Grafiken von 16*16 Pixel für die neue Reitergrafik-Funktion in Forms 12*c* entwerfen.

MDI-Toolbar und Trees brauchen eine Grafik, die links oben zentriert ist. Alle anderen Buttons brauchen zentrierte Grafiken möglichst in 32*32 Pixel. Bei Doppelnutzung (MDI-Toolbar und Maske) müssen Icons in zwei Fassungen erstellt sein. Ein Parallelbetrieb von 6*i* und 12*c* ist sinnvoll, um Risiken eines Produktionsausfalls zu vermeiden. Vielleicht sollte aber die Weiterentwicklung dabei eingefroren werden, sonst wird der Test der neuen Version immer bruchstückhaft und mühselig sein, weil er nie auf einem festen Stand aufsetzen kann – vielleicht auch nie fertig wird.

Konfiguration und Deployment sollten nicht unterschätzt werden, eine Implementation, ob Client oder Server, ist mit WebLogic Servern sehr aufwändig. Wenn Reports weiterentwickelt werden sollen, fällt die schöne Forms-12*c*-Stand-alone-Lösung des Forms-12*c*-Builders weg und es kommt zu wuchtigen und aufwändigen Installationen wie in 11*g*R2 mit mindestens 4 GB zusätzlichem Hauptspeicher.

In Aufruftests müssen die Leitungen zum Forms-Server auf Latenzen geprüft werden. Bei Sicherheitsnetzen ist zu prüfen, ob der lokale DNS-Server die Zertifikate der Java-Applets auflösen kann. Möglicherweise müssen 64-Bit-Java-Clients ausgeschlossen werden – sie könnten instabil sein. Oracle empfiehlt lapidar, bei Problemen mit 64-Bit-Java-Runtime-Umgebungen auf 32-Bit-Java-Versionen zu wechseln.

Nicht jeder Rahmen und jeder Pixel-Abstand wird in der neuen Version „1:1" übertragbar sein. Kleinere manuelle Anpassungen werden später noch erfolgen müssen. Ein zukünftiges Ziel wird in 12*c* auch immer sein, die Netzwerk-Lasten zu vermindern – manchmal ist es sinnvoll, ein Dokument, einen Text oder ein XML-Dokument erst einmal auf dem Server zu generieren, um es dann im Ganzen an den Client zu transportieren. Eine gute Dokumentation hilft, den Migrationsprozess in einem Modul später sauber nachvollziehen zu können.

Reports 12c

Nur zwei Sätze dazu: Reports ist seit der Version 10*g* nicht mehr geändert worden und bekommt von Oracle keine besondere Aufmerksamkeit mehr. Ob es ein Reports 13*c* geben wird, will Oracle nicht verraten – es wird auf jeden Fall definitiv nicht mehr weiterentwickelt. Man vermutet, dass es weiter so vor sich hin dümpeln wird. Die gute Nachricht ist aber, dass die Migration von Reports 6*i* auf Reports 12*c* mit recht wenig Aufwand möglich ist.

Frank Hoffmann
frank.hoffmann@cologne-data.de
http://forms12c.de/demo
@forms12c